Received June 30, 2021, accepted August 7, 2021, date of publication August 18, 2021, date of current version August 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3105914

Training Neural Networks by Enhance Grasshopper Optimization Algorithm for Spam Detection System

SANAA A. A. GHALEB[1,3,4], MUMTAZIMAH MOHAMAD[1], SYED ABDULLAH FADZLI[1], AND WAHEED ALI H. M. GHANEM[2,3,4]

[1]Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Kuala Terengganu 22200, Malaysia
[2]Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu, Kuala Terengganu 21030, Malaysia
[3]Faculty of Engineering, University of Aden, Aden, Yemen
[4]Faculty of Education Aden and Saber, University of Aden, Aden, Yemen

Corresponding authors: Sanaa A. A. Ghaleb (sanaaghaleb.sg@gmail.com) and Waheed Ali H. M. Ghanem (waheedghanem@umt.edu.my)

ABSTRACT A significant negative impact of spam e-mail is not limited only to the serious waste of resources, time, and efforts, but also increases communications overload and cybercrime. Perhaps the most damaging aspect of spam email is that it has become such a major tool for attacks of cross-site scripting, malware infection, phishing, and cross-site request forgery, etc. Due to the nature of the adaptive unsolicited spam, it has been weakening the effect of the previous discovery techniques. This article proposes a new Spam Detection System (SDS) framework, by using a series of six different variants of enhanced Grasshopper Optimization Algorithm (EGOAs), which are investigated and combined with a Multilayer Perceptron (MLP) for the purpose of advanced spam email detection. In this context, the combination of MLP and EGOAs produces Neural Network (NN) models, referred to (EGOAMLPs). The main idea of this research is to use EGOAs to train the MLP to classify the emails as spam and non-spam. These models are applied to SpamBase, SpamAssassin, and UK-2011 Webspam benchmark datasets. In this way, the effectiveness of our models on detecting diverse forms of spam email is evidenced. The results showed that the proposed MLP model trained by EGOAs achieves a higher performance compared to other optimization methods in terms of accuracy, detection rate, and false alarm rate.

INDEX TERMS Metaheuristic, Grasshopper optimization algorithm, classification, artificial neural network, multilayer perceptron, spam email, spam detection system.

I. INTRODUCTION

Email is considered one of the main methods for information dissemination around the world. It is fast, efficient, inexpensive, and easy to use via smartphones, laptops, desktops, and other latest-generation electronic devices that have made emails most popular [1], [2]. Despite the prosperity in the use of other means of communication over the Internet, such as social networks and instant messaging, etc., emails have been kept the leadership in social, academic, and business communications and remain a prerequisite for other means of communication and electronic transactions. The widespread use of e-mail has led to a marked improvement in academic, business, and social communications, which is showing its impact on the growing economy worldwide [3].

Despite the tremendous benefits of using emails, this communication technology has been accompanied by a large number of unsolicited emails and occasionally fraudulent emails, which are collectively known as spam emails. Spam email is among the most annoying Internet phenomena that challenge major global companies, including AOL, Google, Yahoo, and Microsoft [4]. Spam causes many problems which in turn may cause economic losses, e.g., traffic problems and bottlenecks that limit memory space, computing power, and speed. Users also spend large amounts of time trying to remove spam. For these reasons, spam must be detected and isolated immediately through what is referred to as the Spam Detection System (SDS). Spam Detection (SD) is urgently needed to safeguard email users and prohibit

The associate editor coordinating the review of this manuscript and approving it for publication was Chakchai So-In.

many of the negative usages of emails. Unfortunately, due to the adaptive nature of spam emails through the use of mail tools, the effectiveness of SDS has often been limited and sometimes incompetent, hence the need for better SDS to achieve higher accuracies detecting spam and maintaining a low false-positive rate [5].

Five techniques have been widely applied for email spam detection [6]: (1) *Content Based Filtering*, Content-Based Filtering, which is based on analyzing the frequency and distribution of words and phrases in the content of emails and creating automatic filtering rules to classify incoming emails using Machine Learning techniques such as Neural Network (NN), Support Vector Machines (SVM), k-Nearest Neighbor (KNN), and Naïve Bayesian (NB) classification; (2) *Case Base Spam Filtering*, is a common spam filtering method, which is also known as *sample base filtering*. This method uses a collection model to extract all non-spam and spam emails of each user and implement pre-processing to convert the e-mail using feature selection/extraction, dimensionality reduction, then classify the data into two vector sets. The final step in this method uses ML techniques to train and test the datasets in order to determine whether the incoming emails are spam or non-spam [7]; (3) *Heuristic or Rule Based Spam Filtering*, uses previously created rules or inferences to evaluate a large number of patterns that are usually regular expressions against the chosen message. Some score of the message is increased if the known patterns are found. Otherwise, the score of the message is decreased. If the score of the message exceeds a certain threshold it is identified as spam; else, it is considered non-spam. Although some ranking rules do not change over time, other rules need constant updating to be able to effectively deal with the threat of spammers who constantly deliver new spam messages that can easily escape without being noticed by email filters. The SpamAssassin dataset is a popular example of a rule-based spam filter [7], [8]; (4) *Adaptive Spam Filtering*, is mainly based on the use of similarity measurement techniques. It is based on grouping email into different classes used in the detection and filtering of spam. The email body is separated into different groups, each of which has a symbolic text. Each incoming email is compared to each group, and a percentage of similarity is produced to determine the possible group to which the email belongs [9]. Finally, (5) *Previous Likeness Based Spam Filtering,* is based on prior knowledge, so memory-based or instance-based ML is used to classify incoming emails based on their similarity to stored samples (e.g., training emails). This technique uses the kNN for classifying spam emails. A multi-dimensional space vector is created from the email features which are used to plot new instances as points. The new samples are then assigned to the most popular class among the K-closest training instances [10].

All the above techniques are based mainly on two basic approaches: a Machine Learning approach and a knowledge engineering approach [11]. The knowledge engineering approach is called rule-based filtering, which creates a set of rules by using rule-based spam filtering tools or by some other authority [12]. These rules must be maintained continuously and must be updated, which is a waste of time and is inconvenient for most users. In contrast, a Machine Learning approach is called learning-based filtering and is more efficient than a knowledge engineering approach and does not require rules; instead, a set of pre-classified emails is used. Several previous studies have indicated that Machine Learning has played an important role in terms of detecting malicious emails on the Internet, using supervised, unsupervised, and semi-supervised techniques. Supervised Machine Learning algorithms are trained to excerpt the knowledge that can be used to detect spam. This requires a large email dataset with labels (spam and legitimate) to produce an effective classification model. On the contrary, unsupervised Machine Learning algorithms do not rely on labels of the data, rather, it looks for patterns and natural groupings among the instances in the dataset. Semi-supervised algorithms build their models from a dataset that includes both labeled and unlabeled data, usually mostly unlabeled [13].

There are many metaheuristic algorithms that have been improved, the most popular metaheuristic techniques are: Artificial Bee Colony (ABC), which is an inspired improvement method by imitating the behavior of bees, [14], [15]; Particle Swarm Optimization (PSO) [16], which is inspired by behavior of swarms of fishes and birds; Genetic Algorithms (GA) [17], which are inspired by the method of optimization and research, relying on three operators such as selection, crossover, and mutation; Dragonfly Algorithm (DA) [18], which are inspired by dynamic and static swarming behaviors of dragonflies in nature; Cuckoo Search Algorithm (CS) [19], which is inspired by the parasite egg incubating behavior of some cuckoo birds in the nests of other birds; Ant Lion Optimization (ALO) [20], which is inspired by the interaction of ant lions in nature; Monarch Butterfly Optimization (MBO) [21], which is inspired by the migration behavior of monarch butterfly; Bat Algorithm (BAT) [22], which is inspired by the echolocation behavior of bats in nature; Harmony Search (HS) [23], which is inspired by the process of improvising musical harmonies by musicians in an orchestra; Moth-Flame Optimization Algorithm (MFO) [24],which is inspired by the navigation method of moths in nature called transverse orientation; Chicken Swarm Optimization (CSO) [25], which is inspired by the hierarchical system in the chicken swarm that include the roosters, hens, and chicks; Sine Cosine Algorithm (SCA) [26], which is based on sine and cosine mathematical model to generate multiple initial random candidate solutions and requires them to fluctuate outwards or towards the best solution; Salp Swarm Algorithm (SSA) [27], [28], which is inspired by the behavior of salp swarms when foraging in the ocean and navigating; Ant Colony Optimization (ACO) [29], which is inspired by the behavior of ants keeping tracks between colonies and a food source using pheromone; and Animal Migration Optimization (AMO) [30], which is inspired by the behavior of animal swarm migration.

In general, metaheuristic algorithms are enjoying great popularity due to their flexible and powerful performance in dealing with complex real-world problems. They have the ability to exploit/explore the useful information of the population in order to find the optimal solutions. These efficient and durable algorithms are used to tackle various problems such as feature selection problems, single-objective problems, multi-objective problems, engineering design problems, and Machine Learning training problems.

The literature shows that the Machine Learning approach in email filtering attains efficient classification performance. Some of the most popular spam email classification algorithms are Artificial Neural Network (ANN) [31]–[33], SVM [5], and Naive Bayes Classifier [4]. Many studies have also combined Machine Learning algorithms or hybridized more than one algorithm to produce a more accurate and robust detection method [34]. However, these methods still suffer from being time-consuming and very costly. ANNs are commonly used techniques that show accurate classification results of spam email detection [35]–[38] and are inspired by the biological neural system. The most popular applied type of ANN is the Multilayer Perceptron (MLP).

The traditional method used to train connection weights in the MLP model is based on gradient descent technique, such as the Backpropagation algorithm. However, all the gradient based methods suffer some main drawbacks like slow convergence, high dependency on the initial parameters and the high probability of being trapped in local minima [39]–[42]. There are three major drawbacks to an ANN-based SDS.

- The error function of an ANN is a multimodal function that is frequently trapped into local minima.
- This type of ANN-based SDS demonstrates a slow convergence.
- Overly complex models usually result in over-fitting.

In this work, we propose a Machine Learning approach based on MLP classifier trained by a recent metaheuristic optimizer called Grasshopper Optimization Algorithm (GOA). In this approach, which will be referred to as GOAMLP, where the GOA is utilized to optimize the parameters of MLP. The GOA mimics the navigation of adult grasshopper in nature when forming one of the largest swarms on the planet. Some of the main advantages of this algorithm are small numbers of controlling parameters, adaptive exploratory and exploitative search patterns, and gradient-free mechanism. The GOAMLP will be applied on three different spam datasets and compare with other MLPs trained with the other common metaheuristic algorithms, namely, ABC, ALO, BBO, CS, DA, DE, GSA, HS, MBO, MFO, PBIL, PSO, SCA, and WOA. The goal is to enhance the detection accuracy. Therefore, the contribution of this work can be summarized in the following three points:

1. Applying the new enhanced EGOAs algorithms to optimize MLP and address the shortcomings of ANNs in the field of spam detection.
2. Assessing the validity, performance, and reliability of the new technique in detecting new threats by using three datasets (SpamBase, SpamAssassin, and UK-2011 Webspam).
3. Comparing our new proposed model with other evolutionary and swarm intelligence algorithms, using the three spam datasets to confirm the superiority of our model.

The next sections will present the following; section II will present related work. Section III reviews the materials and methods which include: an overview of the proposed SDS, GOA, EGOA, and discusses how GOA can be adapted to train the MLP. Section IV presents the performance evaluation and discussion. Section V discusses the analysis of the experiments used. And finally, section VI discusses the conclusion of the research.

II. RELATED WORK

This section briefly introduces the relevant work that as used to enhance the performance of the GOA by introducing random mutation operators, and then, we proceed to discuss the related works on MLP training using these stochastic techniques.

One of the works that contributed to the enhancement of the GOA is our previous work in [43] on which we depend to train the MLP for SD in the present paper. There have been some earlier endeavors to improve the original GOA with opposition-based learning (OBL) strategy. For instance, OLBGOA has been developed by [44] for solving the global optimization functions and engineering problems. It has two phases, the first phase generates an initial population and its opposite using the OBL strategy, and the second phase uses the OBL as an additional phase to update the GOA population in each iteration. However, in order to reduce the time complexity, the OBL strategy is only utilized for half of the solutions and lacking the required randomness to enhance the convergence rate of the algorithm.

In [45], the authors have proposed IGOA by adding three strategies to the original GOA to solve the global unconstrained and constrained optimization problem. The Gaussian mutation first gives GOA the ability to perform more robust local search, then, using Levy flight strategy, which enables GOA to perform more robust global search. Finally, Opposition-based learning strategy to improve the convergence rate of the algorithm. Still the technique is lacking adequate population diversity and randomness and suffering of extra time complexity than the original GOA. Another limitation of IGOA is that only four functions were used to verify the performance.

In [46], the CGOA algorithm has been proposed incorporating the Chaos theory into GOA in order to accelerate the speed of global convergence. The study integrates 10 of the most widely chaotic maps into GOA: Chebyshey map, Circle map, Gauss/mouse map, Iterative, Logistic map, Piecewise map, Sine map, Singer map, Sinusoidal map, and Tent map, which aim to efficiently balance exploration and exploitation. The results of the study show that the chaotic maps (particularly the circle map) are able to improve the performance of GOA. Nonetheless, the algorithm fails to handle many benchmark functions due to unsuitable chaotic factors.

In [47], yet another improved GOA based on chaos theory (CGOA) has been designed to solve the optimal chiller loading problem. The algorithm key goal is to minimize the energy consumption in chillers by considering the ratio of the part loading in chillers as the optimization parameter. In spite of the improvement the technique has fallen short of tackling the inability to avoid local optima leads which has led to excessively long time in selecting the best chaotic map of the parameter. Another concerning limitation to this approach is only four benchmark functions are used to verify the performance.

In [48], a nonlinear comfort zone parameter/ Levy flight mechanism-based GOA has been suggested. But the algorithm still at times would fall in the trap of local optima after a quick convergence into a promising area. The results also shows that the nonlinear comfort zone parameter and the Levy flight factors were not suitable when handling many optimization functions. The study proposes a new algorithm based on GOA to solve optimization problems and task scheduling problems.

Finally, in [49], the authors propose IGOA. A new enhanced GOA with two modification steps in order to tackle a few of the shortcomings of GOA. The first step is for the variation of the c factor which determines the optimum point more quickly and accurately. The second step provides random walks to the grasshopper in its swarm. However, the technique impairs exploitation capability of IGOA, suffers local optimal trap, and lacks fast convergence towards the optimal values. The lack of good randomness leads to a lack of diversity during the search iterations. And as a result, every search agent could only search the determinate position.

Therefore, in recent years, many researchers proposed to use Stochastic Global Optimization (SGO) methods for training MLPs which are based on generating a number of random solutions to solve the problem. One type of SGO method that is getting more interest in training NNs is the Nature-Inspired Metaheuristic Algorithms (NIMAs). The NIMAs are an example of SGO methods that are becoming more popular in training NNs.

The SGO methods of Swarm Intelligence (SI) can be used to train NNs; they offer an alternative to trajectory driven methods. Trajectory-driven training methods are analogous to an error-minimizing process. The error surfaces generated by NNs are complex surfaces having multiple local minima points and the trajectory-driven paradigm is generally considered to be inefficient in searching for the global minimum of such search space. They tend to converge to a local solution that is near the starting point of search. The basic advantages of the SGO methods to the training of an MLP are their ability to address redundancy in training patterns, the inclusion of training data that are currently not in training set (i.e., the possibility of dynamic learning), and faster training than trajectory-driven training methods. Many SGO methods showed improvements in accuracy and efficiency computation, in comparison with the trajectory-driven methods such as Backpropagation (BP) and Levenberg Marquardt (LM) algorithms. Using such SGO methods for NN training, many problems associated with BP can be overcome [50]. Many algorithms fall under this category, e.g., PSO [51], and ABC [52].

In [35], the authors developed a detection model based on training BPNNs using GA. The algorithm optimizes the weights of the BPNN, improves the classification accuracy, false-positive rate, and false-negative rate. The results were not particularly ideal because, in this study, the authors used private data set to evaluate the model.

In [53] the authors developed an approach where the algorithm of the Negative Selection Algorithm (NSA) is applied to optimize the weights of the BPNN. After the optimization of the BPNN, it is employed into the SD. The algorithm improves the false rate and effective performance, based on the data in a spam dataset.

In [54] the authors proposed an approach for SD based on a feedforward NN classifier that can be used for SD in an offline mode. A Memetic Algorithm (MA), which is an extension of the traditional GA, is employed to optimize the interconnection weights of the NN for SD. Its performance has been evaluated on the UCI SpamBase dataset. However, the MA traditional GA was limited because of its lack of local optimal.

In [55] the authors presented an SD using a feedforward NN. The network is trained by a new optimization technique that was inspired by the herding behavior of real small crustaceans called Krill Herd Algorithm (KH), to identify spam and non-spam. The ANN trained by KH showed higher accuracy and faster convergence speed compared to the traditional PB algorithm. However, the limitation of KH-FFNN is that only one dataset was used to verify the performance.

In [56] have proposed SD based on ANN trained by Biogeography Based Optimization (BBO) algorithm. Their model has demonstrated a high accuracy of identifying spam e-mails compared to the other approaches. Its performance has been evaluated on the SpamAssassin and SpamBase dataset.

In [57] the authors proposed a new SD based on an ANN trained by the enhanced Bat Algorithm (EBAT) to learn the patterns of spam email given in SpamBase and UK-2011 Webspam training datasets. In addition to the aforementioned works, we propose a new SDS model built using the most promising GOA algorithm to train the MLP in solving the problems encountered by the traditional MLP's training algorithms. Our proposed model is able to detect spam email and false alarm rates in SpamBase,

SpamAssassin, and UK-2011 Webspam benchmark datasets with higher accuracy. The UK-2011 Webspam dataset contains a new emerging attack compared with the SpamBase and SpamAssassin datasets. Our experiments have shown that our new SDS model performs better than all the other techniques in the literature.

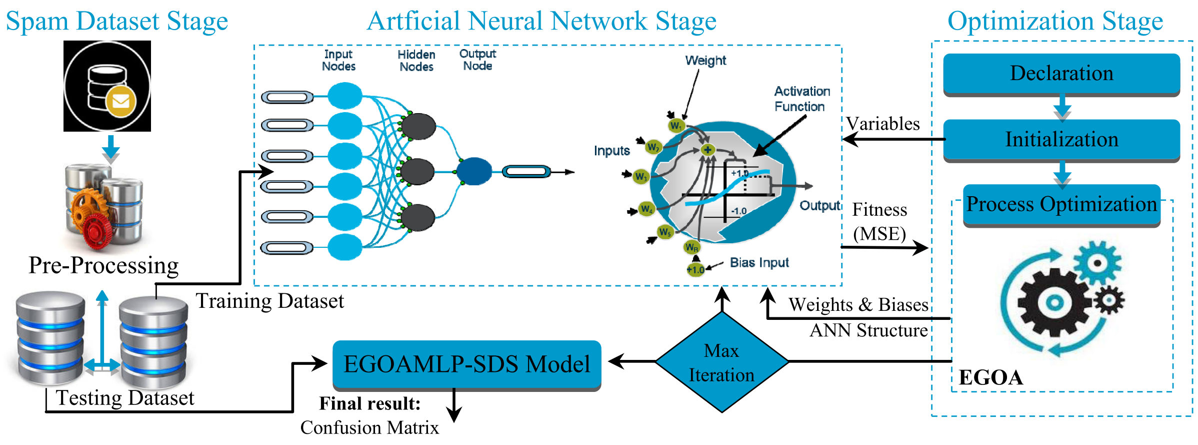

FIGURE 1. **THE EGOAMLP-SDS Model.**

In [4] the authors proposed an approach for SDS, using Water Cycle Feature Selection (WCFS) of the spam message and used three ways of multiple classifiers algorithms which are SVM, KNN, NB. Its performance has been evaluated on the benchmarks, SpamBase. However, the performance can be improved by using different datasets combinations of features.

In [58], the authors proposed an approach for SDS, using a Machine Learning -based Content analysis of the spam message. And the system is composed of six modules. Which two different spam detectors were tested, the former with Synonym Replacer (SR), the latter without synonym replacer. Its performance has been evaluated on the benchmarks, SpamAssassin, Trec2007. However, the results obtained are poor.

In [59] the authors developed an approach for BP with momentum optimized ANN and BP optimized ANN. Its performance has been evaluated on the benchmarks, SpamBase. However, BP and BP with momentum are very popular, it gets trapped in local optima.

Finally, in [50] the authors proposed an approach for SD based on a feature-centric spam email detection model based on content, sentiment, semantic, user, and spam-lexicon features set. Its performance has been evaluated on the benchmarks, SpamAssassin. However, more efficient results can be obtained by introducing more features.

III. METHODOLOGY OF THE STUDY

In this section, the methodology of this study designs and implements the SDS model based on an ANN, which is trained by the EGOAs algorithms, as shown in Figure 1. Our purpose is to design a new SDS model that achieves promising scores in terms of classification accuracy, detection rate, false alarm rate, global convergence and exhibits strong robustness in identifying spam emails with the help of a series of six different variants of enhanced Grasshopper Optimization Algorithm (EGOAs) for training the ANNs.

MLPs are powerful classification tools that have proven highly capable of solving the challenges faced by SDS. This kind of tool recognizes the typical characteristics of the system users, the context of electronic mailing, and statistically significant deviations from established user behavior. These tools have open and extendable structures, which help them create a general knowledge model of the behavior of an environment. The procedure chiefly entails determining the structure and parameters of the NN to learn the relationship between the incoming patterns and the target output by training.

The MLP training is a complex task of prominent importance in problems that require learning. This training process can be classified as an optimization problem. The training procedure can derive its solution from a linear constraint with a nonlinear optimization problem, therefore, different optimization algorithms are applied in the literature to solve this problem. The enhanced GOA algorithm is a stochastic global optimization that aims to find a convenient solution or approximate optimal solutions in a search space. EGOAs addresses the problem of trapping in a local optimum solution in the search space before reaching the global optimum with very fast convergence; it has powerful robustness and global convergence. Figure 1 illustrates the model of the proposed SDS and shows that it can be divided into several modules, namely, the spam dataset module, ANN module and the optimization module, which can be described as follows:

The first stage in the proposed model is using spam dataset module (*SD module*) in order to process, filter, and extract the features from the audit data. The dataset contains predefined training and testing sets that are used as inputs for the next ANN module. Before entering data into the NN module, the input data should be normalized to fall between $[-1, +1]$, to render these data usable for the next module.

The second stage uses the ANN module, which receives the training features of the input data from the spam dataset module. The ANN module is designed as a MLP, which consists of one input layer, a number of hidden layers, a number of neurons in each hidden layer, and one output layer. The outbound data from the SDS module (the training dataset) are fed into the ANN module as an input training pattern for training the ANN. This training process is performed using the error evaluation criteria of Mean Squared Error (MSE) as input to the optimization module (the EGOA module) and receiving the updated structure and weights/biases of the network from the EGOA module.

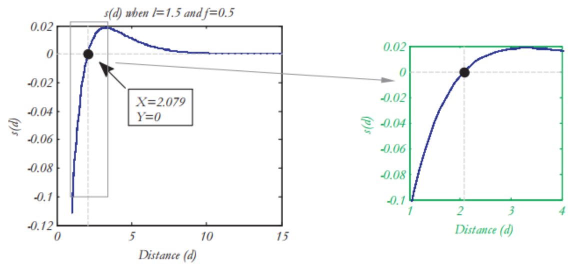

FIGURE 2. Function S when L = 1.5 and F = 0.5, and range of function S when X is in [1], [4] [61].

The EGOA module is developed as a standalone system for updating the structure solution and weights/biases solution after each iteration. In each iteration of the training process, the EGOA module sends its solutions as a set of structure and weights/biases into the ANN module, which assesses these solutions based on a training dataset and then returns their fitness values. In this study, the MSE is selected as a known fitness function for the proposed EGOA training algorithm. The structure and weights/biases for the ANN are obtained by minimizing the value of MSE. Afterward, the knowledge base (ANN structure and weights/biases) is updated. The maximum number of iterations parameter shown in Figure 1 (Max Iteration) controls the stopping of the training process. The last stage is carried out after obtaining the best model in terms of the best ANN structure and weights/biases, which was built by using the training dataset. Here, the testing inputs are fed from the testing dataset into the trained ANN to predict the output. The testing process of the ANN can be viewed as checking the predicted output with the closest match to any of the target classes.

A. THE GRASSHOPPER OPTIMIZATION ALGORITHM

The Grasshopper Optimization Algorithm (GOA) is one of the metaheuristic algorithms that were inspired by the behavior of grasshopper insects as proposed in 2017 by Saremi et al [61]. GOA is also faced with the problem of being trapped in local optima and slow convergence. These disadvantages limit the wider application of GOA. Although grasshoppers are usually seen individually in nature, they constitute the largest swarms of all insects. A large number of these insects produce a swarm of pest that causes severe damage to crop production and agriculture; a large number of grasshoppers is a nightmare for farmers. The life cycle of the grasshopper swarms consists of two stages: nymphs and adults.

The nymph grasshopper moves slowly over a small distance, which helps to exploit their living area and eat all the plants on their paths. On the other hand, the adult grasshopper has two main tasks: finding food and migration. It can jump high and move over a large distance to find food and therefore have a larger area to explore. We can conclude that both movements carried out by the grasshopper, that is, slow movement (small distance) and sudden movement (large distance) of the large group of swarms, corresponds to exploration and exploitation. In exploration, the grasshoppers (finding food) tend to move a large distance, whilst they prefer to move locally during the exploitation stage. These two tasks, as well as finding a food source, are achieved by grasshoppers naturally. The mathematical model that represents the swarming behavior of the grasshopper was offered in [61], and is repeated here:

$$X_i = S_i + G_i + A_i, \tag{1}$$

where X_i represents the position of the i^{th} grasshopper, S_i represents the social interaction given in Eq. (2), G_i represents the gravity force on the i^{th} grasshopper, and A_i represents the wind advection. The social interaction S_i is given by:

$$S_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} s(d_{ij})\widehat{d_{ij}} \tag{2}$$

where d_{ij} represents the distance between the i^{th} and j^{th} grasshoppers, and is calculated as $d_{ij} = |x_i - x_j|$, parameter s is a function to calculate the strength of social forces, which is calculated by Eq. (3), and $\widehat{d_{ij}} = |x_i - x_j| / d_{ij}$ represent the unit vector from the i^{th} grasshopper to the j^{th} grasshopper.

$$s(r) = fe^{r/l} - e^{-r} \tag{3}$$

With f is the intensity of attraction and l represents the attractive length scale. Grasshoppers create three food-seeking regions in terms of social interaction: the comfort zone, the repulsion region, and the attraction region. The function s is illustrated in Figure 2 to show elicits forces of attraction and repulsion between the grasshoppers. One can observe from the figure that when the distance is in the range [0, 2.079], the grasshoppers repel each other to avoid impact. However, there is no attraction nor repulsion when the difference is exactly at 2.079, and agents are said to be in the comfort zone. As the distance extends beyond 2.079, the function keeps increasing till the distance reaches a value of around 4. This range [2.079, 4] is called the attraction phases. Here, the grasshoppers cooperate to reach the food source. Different values of f and l would give variant zones of repulsion, comfort, and attraction. However, for the GOA, the values of $f = 0.5$ and $l = 1.5$ are used. Despite the excellent modeling of the different zones by Eq. (3), it gives a value of almost zero when the distance goes beyond 10 (Figure 2). Thus, to solve this issue, the distances between the agents are projected to the range [1, 4]. The G_i component in Eq. (4), which is calculated as follows:

$$G_i = -g\hat{e}_g \tag{4}$$

where g represents the gravitational constant whilst $\hat{e}_g$ is the unity vector toward the center of earth. The last argument A_i in Eq. (1) is calculated by Eq. (5).

$$A_i = u\widehat{e_w} \tag{5}$$

From Eq. (5), u is constant drift and $\widehat{e_w}$ is the unity vector in the direction of wind. The nymph grasshopper has no wings, so their movements are closely related to the wind direction. After substituting Eq. (2), Eq. (4), and Eq. (5) in Eq. (1), the final equation becomes:

$$X_i = \sum\nolimits_{j=1,j\neq i}^{N} s\left(\left|x_j - x_i\right|\right)\frac{x_j - x_i}{d_{ij}} - g\hat{e}_g + u\widehat{e_w} \tag{6}$$

where $s(r) = fe^{/l} - e^{-r}$ and argument N in Eq. (6) represents the number of grasshoppers. Because nymph grasshoppers land on the ground, their position should not go below a threshold. However, Saremi *et al.* were unable to utilize Eq. (6) for modeling any swarm simulation or optimization algorithms because it hinders the algorithms from exploiting and exploring the search space close to the solution.

The nymph grasshopper algorithm is designed for a grasshopper swarm which stays in free space. Furthermore, the mathematical Eq. (6) cannot be used directly to solve optimization problems, as the grasshoppers rapidly achieve the comfort zone, and the swarm does not converge to a specified point. A modified version of Eq. (6) by Eq. (7) was proposed by Saremi et al [61] to solve optimization problems:

$$X_i^d = c\left(\sum\nolimits_{j=1,j\neq i}^{N} c\frac{ub_d - lb_d}{2} s\left(\left|x_j^d - x_i^d\right|\frac{x_j - x_i}{d_{ij}}\right)\right) + \widehat{T_d} \tag{7}$$

In Eq. (7) the ub_d *and* lb_d parameters represent the upper and lower bounds in the D^{th} dimension, respectively, while $s(r) = fe^{r/l} - e^{-r}.\widehat{T_d}$ represents the best solution value of the D^{th} dimension in the current iteration, and argument c represents the decreasing coefficient to detract from the comfort area, attraction area, and repulsion area. It aims to strike a balance between exploration and exploitation, as after the arrival of grasshoppers to the promising areas it needs to reduce exploration and increase exploitation, by obliging the grasshopper to search locally to find an accurate approximation of the global optimum. Therefore, the argument c is required to be decreased proportional to the number of iterations. This mechanism encourages exploitation as the iteration count increases. The argument c reduces the comfort zone proportional to the number of iterations and is calculated as follows:

$$c = c_{max - Iter\frac{c_{max} - c_{min}}{iter_{max}}} \tag{8}$$

It should be noted from Eq. (7) that the coefficient c plays two main roles: The task of the first coefficient c which is outside the main brackets is quite similar to the inertial weight (w) in PSO. It decreases the grasshopper movements around the optimum solution. In other words, this coefficient balances the exploitation and exploration of the entire swarm around the optimum solution. While the second coefficient c decrease the distances of the three regions (comfort zone, repulsion zone, and attraction zone) between grasshoppers. Also, the component $[c\frac{ub_d - lb_d}{2}]$ is linearly decreases the space that the grasshoppers should explore and exploit. The second part $s\left(\left|x_j - x_i\right|\right)$ indicates if a grasshopper should be repelled from (exploring) or attracted to (exploiting) the target. Parameter c_{max} represents the maximum value, c_{min} represents the minimum value, $Iter$ is the current iteration, and $iter_{max}$ indicates the maximum number of iterations.

In original work that proposed the GOA algorithm, the authors used the values 1 and 0.00001 for c_{max} and c_{min}, respectively. The next grasshopper position is computed based on its current position, all other grasshoppers' positions, and the best grasshopper's position obtained so far, as given in Eq. (7). Note that the first component of this equation considers the location of the current grasshopper with respect to other grasshoppers. The GOA is using only one position vector for every search agent. This means that GOA updates the position of a grasshopper based on the global best, its current position, and all other grasshoppers' positions in search agents. Thus, GOA requires all search agents to get involved in defining the next position of each search agent.

Algorithm (1) lists the pseudo code of the GOA algorithm. As shown in the algorithm 1, the first step is to initialize all the parameters such as the maximum number of iterations, the maximum value of coefficient c, the minimum value of coefficient c, and the number of populations. The GOA algorithm starts by generating a random population $x_i^d, i = 1, 2, \ldots, N, d = 1, 2, \ldots, dim$, (an array of N rows and dim columns) and calculating the fitness function for each

Algorithm 1 Grasshopper optimization algorithm (GOA)

```
*Initialize all the parameters such as:*
*Maximum No. of iterations ($iter_{max}$), $c_{max}$, $c_{min}$, and number of population ($N$);*
*Generate a random population ($X_i^d$), $i = 1, 2, 3 \ldots, N$; and $d = 1, 2, \ldots.$ Dim (no. of dimensions);*
*Calculate the fitness of each grasshopper;*
$\hat{T_d}$ = *the best grasshopper;*
**While** ($iter < iter_{max}$)
  *Update the parameter c using Eq. (8);*
  **for** *each grasshopper in population*
    *Normalize the distances between grasshoppers in $X_i^d$ to [1], [4];*
    *Update $x \in X_i^d$ by using Eq. (7);*
    *Adjust the boundaries for the current grasshopper in population;*
  **end for**
*Update **T** if there is a better solution;*
$iter = iter + 1$;
**end while**
**Return** *the best solution of **T**;*
```

grasshopper (solution) $x_d, d = 1, 2, \ldots, dim$, and the best grasshopper (solution), T_d, is then selected according to the best fitness function. Thereafter, for each grasshopper x_i of the population, the following three steps are implemented: 1) Normalizing the distances between grasshoppers in X_i^d to [1], [4]; 2) Update the current grasshopper $x_i \in X_i^d$ by using Eq. (7); 3) Adjusting the boundaries for the current grasshopper in the population. In the last step, the grasshopper position updates are made frequently until the end criterion is met.

The grasshopper position and fitness of the best target is finally returned as the best approximation for the global optimum. In real-world engineering applications, GOA has been applied to solve many optimization problems including linear antenna arrays, function optimization, flow shop scheduling, reliability problem, economic load dispatch, and others [62].

B. THE ENHANCED GOA ALGORITHM

As discussed above, the original GOA algorithm has the ability to exploit the search space, but it occasionally fails into local optima trap, which adversely impacts the overall performance. Therefore, in this section, we are presenting six enhanced variations of the GOA (EGOAs). This has been carried out by integrating several of the mutation operator's strategy in order to improve the ability to deeply explore and exploit the search space and swiftly reach the optimal value. The new six proposed algorithms are E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA; and their detailed pseudocodes are shown in Algorithm 2, Algorithm 3, Algorithm 4, Algorithm 5, Algorithm 6, and Algorithm 7 respectively.

In order to enhance the diversity of the solution randomization would be a required component to GOA. Randomization helps the algorithm to explore multifarious regions with a high solution diversity. GOA has coefficient c which has a similar role, but it is confined to certain local jumps. Additionally, the mutation operator, which happens to be the probabilistic operator too, randomly modifies grasshopper movement in search spaces based on the grasshopper prior probability of existence.

Aiming to allow the original search space to reach the optimal solution swiftly and renovate the out-of-range solutions, the six aforementioned mutation operator's strategies are only applied to the GOA's exploitation phase in order to maintain 50% of the domain space that has been calculated by GOA. Generally, each algorithm would consist of two main phases, initial, and updating phase. In order to decide which type of search phase to adopt the algorithms examines the aging degree of probability values for each individual.

1) E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, AND E6GOA ALGORITHMS

In this section, we propose six enhancements strategies to the GOA algorithm. First is E1GOA which is based on the original GOA, but it aims to enhance the performance of GOA by using a highly efficient mutation operator. To help improving the diversity of the solution as well as the convergence to the optimal value the operator is taken as a set of random-based modifications which would also allow for more mutations in solutions that have been examined in grasshopper search whilst assisting to avoid local optimum traps. This provides the algorithm with the ability to exploit solution in the local neighborhood whilst exploring new areas in the search space, two crucial characteristics of an effective optimization algorithm [63].

E1GOA algorithm has three stages, initialization, updating, and final phase. It is worth mentioning that not only does E1GOA adopts all the parameters of the original GOA, but it also adds four new control parameters p, *limit1, limit2*, and w. In the initialization phase, the search space solutions are spotted, and values are assigned to several parameters. During updating phase, E1GOA iterations are performed to modify solutions until a stopping criterion is met. The updating phase consists of two parts GOA phase and the Mutation Operator phase. In each iteration the grasshopper is calculated as such if rand value (ε_1) is less than p, it will perform the classical GOA algorithm (GOA phase); otherwise, the new mutation operator proposed will take place; the ε_1 represents a random number in [0, 1].

The p in E1GOA functions similarly to coefficient p in migration operator and butterfly adjusting operator of the MBO algorithm (the value of $P \in [0, 1]$) [64]. In fact, p is considered a switching parameter to decide the operator to employ either the GOA phase or the new mutation operator phase. For instance, if p is near 1 (*too high*), more solutions will be generated using GOA phase, whereas if p is near 0 (*too low*), more solutions will be generated by using the new mutation operator phase. This decides whether the new mutation operator phase, or GOA phase, will play a most important role in the newly generated solutions.

As described in Algorithm 2, the mutation operator algorithm has three feasible choices whilst evaluating newly generated grasshopper: (1) grasshopper is updated based on the randomly selected grasshopper solution vectors, i.e., it is choosing a solution from previously improved solutions. (2) The newly generated grasshopper is updated close to the best-known solutions, or (3) the newly generated grasshopper is updated from a feasible range. When E1GOA selects a value of a decision variable from solution vectors, the value is modified using these three rules. The grasshopper position could be optimized using three factors: solution vectors, control parameters (*limit1* and *limit2*), and randomization. If the newly generated grasshopper has a significantly improved fitness, then it replaces the worst grasshopper in the solution vectors. The updating process is carried out until a stopping condition is fulfilled.

The new mutation operator proposed in this work includes two control parameters, *limit1* and *limit2* $\in [0, 1]$. These two parameters play a similar role to the Harmony Memory Consideration Rate (HMCR) and the Pitch Adjustment Rate (PAR) used in the Harmony Search Algorithm (HS) [65], which is inspired by the musical improving process.

If the value of the parameter *limit1* is excessively high (*close to 1*), then the majority of the grasshopper solutions are used in solution vectors; as a result, other grasshopper solutions are not explored well, thereby leading to incorrect solutions. If the value is extremely low (*close to 0*), then only a small number of fittest grasshopper solutions are chosen, which then leads to a slow convergence rate. As a rule, limit1 ranges from 0.7 to 0.95.

An appropriate method needs to be used to adjust the grasshopper movement successfully and thus adopt the jump significantly in the second part. In terms of principle, the grasshopper movement can be slightly adjusted in the linear or nonlinear form theoretically. Linear adjustment is most popular utilized than nonlinearity adjustment. Consequently, the grasshopper movement is updated by Eq. (9) or Eq. (10).

$$x_i^{t+1} = w \times \left(x_{ir_2}^t - x_{best}^t\right) \times 2 \times (rand - 1); \quad (9)$$

$$x_i^{t+1} = w \times \left(x_{ir_1}^t - x_{best}^t\right) \times 2 \times (rand - 1); \quad (10)$$

In the equations above, x_i^{t+1} is a new position of grasshopper i at iteration $t+1$. In the same way, $x_{ir_1}^t$ represents the newly generated position of the grasshopper r_1. Similarly, $x_{ir_2}^t$ describes the newly generated position of the grasshopper r_2. Moreover, the grasshopper r_1 and r_2 are randomly selected from the population (*Solution Vectors*). If $\varepsilon_3 < limit_2$, then the newly generated grasshopper is updated using Eq. (9). On the other hand, if $\varepsilon_3 > limit_2$, then the newly generated grasshopper is updated using Eq. (10). x_{best}^t represents the best solution ever found at iteration t. whereas *rand* is a random number generated from a uniform distribution in [0, 1]. The variable w in Eq. (9) and Eq. (10) is similar to the inertial weight (w) in PSO algorithm, it reduces the grasshopper movements around the optimum value. This variable plays an important role in balancing exploitation and exploration of the mutation operator around the optimum value.

The *limit2* parameter is very much like the mutation operator in evolutionary algorithms and pitch adjustment rate in the harmony search algorithm. Meanwhile, the *limit2* entirely determines the degree of the grasshopper movement and should be set carefully. If the value of *limit2* is close to 1 (very high), then the solutions significantly change, and the algorithm may not converge at all. If the value of limit2 is close to 0 (very low), then the solutions slightly change, and the algorithm may be premature. Usually, *limit2* ranges from 0.1 to 0.5.

Finally, the randomization in the third element aims to increase the diversity of solutions. Although *limit2* has similar functionality, its functionality is limited to a local location that corresponds to local search. The use of randomization often commences the search process to explore diversified areas to find the global optimal solution.

The main contribution in the proposed algorithm (E1GOA) is to plug the new mutation operator into the GOA algorithm to increase the diversity of the population in an endeavor to enhance the performance of GOA to speed up the convergence to the optimal solution.

Algorithm 2 (E1GOA)

```
Initialize all the parameters such as:
Maximum No. of iterations ($iter_{max}$), $c_{max}$, $c_{min}$, and number of population ($N$);
Generate a random population ($X_i^d$) :
($i = 1, 2, 3 \ldots, N$) and ($d = 1, 2, \ldots$. dim ➜ no. of dimensions);
Calculate the fitness of each grasshopper;
$\hat{T}_d$ = the best grasshopper;
While ($iter < iter_{max}$)
  Update the parameter c using Eq. (8);
  for $i = 1$ to $N$ (grasshopper in population) do
    if $\varepsilon_1 \le p$ then
      GOA phase ();
    else
      if $\varepsilon_2 \le limit_1$ then
        Randomly select a grasshopper in population ($r_1$);
        $x_{ir_1}^{t+1} = x_{r_1}^t$;
        if $\varepsilon_3 < limit_2$ then
          Randomly select a grasshopper in population ($r_2$);
          $x_{ir_2}^{t+1} = x_{r_2}^t$;
          $x_i^{t+1} = w \times \left(x_{ir_2}^t - x_{best}^t\right) \times 2 \times (rand - 1)$;
        else
          $x_i^{t+1} = w \times \left(x_{ir_1}^t - x_{best}^t\right) \times 2 \times (rand - 1)$;
        end if
      else
        $x_i^{t+1} = x_{min}^t + rand \times (x_{max}^t - x_{min}^t)$;
      end if
    end if
  end for
Update T if there is a better solution;
$iter = iter + 1$;
end while
Return the best solution of T;
```

Now we offer the second proposed algorithm, which is E2GOA algorithm, as shown in Algorithm 3. The goal of this algorithm is very similar to the previous algorithm (E1GOA), but the new mutation operator of E2GOA has been slightly modified by comparison with E1GOA to test and evaluate the different forms of mutation operators. We have to take into consideration that these various new mutation operators are possible to give us a different result if it is applied to the same real-world problem. So, we present six different forms of mutation operators in this article. The idea behind E2GOA algorithm is to develop the E1GOA mutation operators by introducing a slight modification to the control parameter. E2GOA has two control parameters: *p* and *limit1*. Parameter *p* in E2GOA plays the same role as parameter *p* in E1GOA. In terms of parameter *limit1*, if $\varepsilon_2 < limit_1$, an individual $x_{r_1}^t$ is chosen from grasshoppers in the population as demonstrated in Eq. (11), and then a new grasshopper x_i^{t+1}

Algorithm 3 (E2GOA)

```
Initialize all the parameters such as:
Maximum No. of iterations ($iter_{max}$), $c_{max}$, $c_{min}$, and number of population ($N$);
Generate a random population ($X_i^d$) :
($i = 1, 2, 3 \ldots, N$) and ($d = 1, 2, \ldots$. dim ➜ no. of dimensions);
Calculate the fitness of each grasshopper;
$\hat{T}_d$ = the best grasshopper;
While ($Iter < iter_{max}$)
  Update the parameter c using Eq. (8);
  for $i = 1$ to N (grasshopper in population) do
    if $\varepsilon_1 \leq p$ then
      GOA phase ();
    else
      if $\varepsilon_2 < limit_1$ then
        Randomly select a grasshopper in population ($r_1$);
        $x_{ir_1}^{t+1} = x_{r_1}^t$;
        $x_i^{t+1} = w \times \left(x_{ir_1}^t - x_{best}^t\right) \times 2 \times (rand - 1)$;
      end if
    end if
  end for
Update T if there is a better solution;
$iter = iter + 1$;
end while
Return the best solution of T;
```

is updated by using Eq. (10).

$$x_{ir_1}^{t+1} = x_{r_1}^t \tag{11}$$

The E3GOA algorithm is the third algorithm proposed in this work and is shown in Algorithm 4. This algorithm is similar to the previous proposed algorithms but with a slight modification in the mutation operator. The mutation operator in E3GOA algorithm is using two control parameters: p and limit1. Here, the variable p plays the same role as in the previously proposed algorithms. And ε_1 and ε_2 in [0, 1] are two random numbers drawn from the uniform distribution. If $\varepsilon_1 \leq p$, the new solution will be generated by using the GOA phase. If $\varepsilon_1 > p$, the new solution will be generated by using the mutation operator. The new mutation operator has one control parameter, *limit1*, such that if $\varepsilon_2 \leq limit_1$, two individual grasshoppers, $x_{r_1}^t$ and $x_{r_2}^t$ are randomly chosen from the grasshopper population. Here N is population size, (r_1) and (r_2) are integer numbers in [1, N]. If (r_1) is not equal to (r_2), then x_i^{t+1} is updated by Eq. (9) else x_i^{t+1} is updated by Eq. (10). However, if $\varepsilon_2 > limit_1$, x_i^{t+1} is updated randomly from the feasible range. x_{best}^t represents the current best grasshopper in the population, t is the current generation number. Variable rand is a random number generated from a uniform distribution in [0, 1].

The E4GOA algorithm is the fourth proposed algorithm in this work and is shown in Algorithm 5. This algorithm is similar to the previously proposed algorithms but with a slight

Algorithm 4 (E3GOA)

```
Initialize all the parameters such as:
Maximum No. of iterations ($iter_{max}$), $c_{max}$, $c_{min}$, and number of population ($N$);
Generate a random population ($X_i^d$) :
($i = 1, 2, 3 \ldots, N$) and ($d = 1, 2, \ldots$. dim ➜ no. of dimensions);
Calculate the fitness of each grasshopper;
$\hat{T}_d$ = the best grasshopper;
While ($iter < iter_{max}$)
  Update the parameter c using Eq. (8);
  for $i = 1$ to N (grasshopper in population) do
    if $\varepsilon_1 \leq p$ then
      GOA phase ();
    else
      if $\varepsilon_2 \leq limit_1$ then
        Randomly select a grasshopper in population ($r_1$);
        $x_{ir_1}^{t+1} = x_{r_1}^t$;
        Randomly select a grasshopper in population ($r_2$);
        $x_{ir_2}^{t+1} = x_{r_2}^t$;
        if ($r_1 \neq r_2$) then
          $x_i^{t+1} = w \times \left(x_{ir_1}^t - x_{best}^t\right) \times 2 \times (rand - 1)$;
        else
          $x_i^{t+1} = w \times \left(x_{ir_2}^t - x_{best}^t\right) \times 2 \times (rand - 1)$;
        end if
      else
        $x_i^{t+1} = x_{min}^t + rand \times (x_{max}^t - x_{min}^t)$;
      end if
    end if
  end for
Update T if there is a better solution;
$iter = iter + 1$;
end while
Return the best solution of T;
```

modification in the mutation operator. While the structure of this algorithm is quite similar to the previous algorithms, the fundamental difference is the method of updating the movement of the grasshopper, which is implemented by Eq. (12) or Eq. (13)., x_{worst}^t represents the worst current grasshopper in the population, and t is the current generation number.

$$x_i^{t+1} = (w \times x_{ir_1}^t - x_{best}^t) \times 2 \times (rand - 1) \tag{12}$$

$$x_i^{t+1} = (x_{worst}^t - w \times x_{ir_2}^t) \times 2 \times (rand - 1) \tag{13}$$

The E5GOA algorithm is the fifth proposed algorithm in this work and is shown in Algorithm 6. This algorithm is similar to the E2GOA algorithm but with a slight modification in the mutation operator, where the update to the movement of the grasshopper is performed using Eq. (12) or Eq. (13).

The E6GOA algorithm is the sixth proposed algorithm in this work and is shown in Algorithm 7. This algorithm is similar to the E1GOA algorithm, but the mutation operator used to update the movement of the grasshopper in search

Algorithm 5 (E4GOA)

```
Initialize all the parameters such as:
Maximum No. of iterations ($iter_{max}$), $c_{max}$, $c_{min}$, and number of population ($N$);
Generate a random population ($X_i^d$) :
($i = 1, 2, 3 \ldots, N$) and ($d = 1, 2, \ldots.$ dim ➜ no. of dimensions);
Calculate the fitness of each grasshopper;
$\hat{T_d}$ = the best grasshopper;
While ($iter < iter_{max}$)
  Update the parameter c using Eq. (8);
  for $i = 1$ to N (grasshopper in population) do
    if $\varepsilon_1 \leq p$ then
      GOA phase ();
    else
      if $\varepsilon_2 \leq limit_1$ then
        Randomly select a grasshopper in population ($r_1$);
        $x_{ir_1}^{t+1} = x_{r_1}^t$;
        Randomly select a grasshopper in population ($r_2$);
        $x_{ir_2}^{t+1} = x_{r_2}^t$;
        if ($r_1 \neq r_2$) then
          $x_i^{t+1} = (w \times x_{ir_1}^t - x_{best}^t) \times 2 \times (rand - 1)$ ;
        else
          $x_i^{t+1} = (x_{worst}^t - w \times x_{ir_2}^t) \times 2 \times (rand - 1)$ ;
        end if
      else
        $x_i^{t+1} = x_{min}^t + rand \times (x_{max}^t - x_{min}^t)$;
      end if
    end if
  end for
Update T if there is a better solution;
$iter = iter + 1$;
end while
Return the best solution of T;
```

space is based on Eq. (14) or Eq. (15).

$$x_i^{t+1} = x_i^t - w \times (x_{ir_2}^t - x_{best}^t) \times 2 \times (rand - 1) \quad (14)$$

$$x_i^{t+1} = x_i^t - w \times (x_{ir_1}^t - x_{best}^t) \times 2 \times (rand - 1) \quad (15)$$

2) ALGORITHM COMPLEXITY ANALYSIS

The computational complexity of the enhanced grasshopper optimization algorithm mainly depends on the number of solutions represented by the dimension (d), and the number of the populations represented by the population size (n). Considering the worst scenario, the overall computational complexity is O (dn) ≈ O (O (calculate the oppositional position of all solutions and evaluate its fitness) + O (sort solutions of population and oppositional population)). In the iterative process of the proposed algorithms (EGOAs), the time complexity of an iteration is analyzed as follows: In step 1, the main operation for produce the initial population, and the time complexity is O (dn). In step 2, judging the stopping criteria, the time complexity is O (1). In step 3, judging the parameter value rand, and if rand is less than the parameter (p), perform the GOA phase the time complexity is O (dn). Otherwise, perform the EGOA phase; by, performing updating the solution of search space, the time complexity is O (d). In step 4, iteration to continue and return to step 3. Therefore, the final computational complexity of the proposed algorithms is O (EGOAs) ≈ O (dn).

Algorithm 6 (E5GOA)

```
Initialize all the parameters such as:
Maximum No. of iterations ($iter_{max}$), $c_{max}$, $c_{min}$, and number of population ($N$);
Generate a random population ($X_i^d$) :
($i = 1, 2, 3 \ldots, N$) and ($d = 1, 2, \ldots.$ dim ➜ no. of dimensions);
Calculate the fitness of each grasshopper;
$\hat{T_d}$ = the best grasshopper;
While (Iter < $iter_{max}$)
  Update the parameter c using Eq. (8);
  for $i = 1$ to N (grasshopper in population) do
    if $\varepsilon_1 \leq p$ then
      GOA phase ();
    else
      Randomly select a grasshopper in population ($r_1$);
      $x_{ir_1}^{t+1} = x_{r_1}^t$;
      if $\varepsilon_2 \leq limit_1$ then
        $x_i^{t+1} = (w \times x_{ir_1}^t - x_{best}^t) \times 2 \times (rand - 1)$ ;
      else
        $x_i^{t+1} = (x_{worst}^t - w \times x_{ir_1}^t) \times 2 \times (rand - 1)$ ;
      end if
    end if
  end for
Update T if there is a better solution;
$iter = iter + 1$;
end while
Return the best solution of T;
```

C. THE GOA ADAPTATION PROCESS

The process of adapting metaheuristic algorithms for the purpose of training an ANN is an important process. In the metaheuristic-based methods, the training process translates to using suitable ANN structure, ANN weights and biases representation, fitness function and termination condition(s). Consequently, for adapting EGOAs as an ANN training method, these aforementioned four aspects can be adapted to suit the functionality of the EGOAs algorithm and satisfy the requirements mandated by the ANN training process in general, thus the new training method is known as EGOAMLP algorithm. The efficiency of the hybridization of the grasshopper algorithm with the NN has already been evaluated and compared with some other meta-heuristic methods and showed promising results [66], which is the main motivation in using the grasshopper algorithm to build our new model.

Algorithm 7 (E6GOA)

```
Initialize all the parameters such as:
Maximum No. of iterations ($iter_{max}$), $c_{max}$, $c_{min}$, and number of population ($N$);
Generate a random population ($X_i^d$) :
($i = 1, 2, 3 \ldots, N$) and ($d = 1, 2, \ldots.$ dim ➜ no. of dimensions);
Calculate the fitness of each grasshopper;
$\hat{T}_d$ = the best grasshopper;
While ($iter < iter_{max}$)
  Update the parameter c using Eq. (8);
  for $i = 1$ to N (grasshopper in population) do
    if $\varepsilon_1 \leq p$ then
      GOA phase ();
    else
      if $\varepsilon_2 \leq limit_1$ then
        Randomly select a grasshopper in population ($r_1$);
        $x_{ir_1}^{t+1} = x_{r_1}^t$;
        if $\varepsilon_3 < limit_2$ then
          Randomly select a grasshopper in population ($r_2$);
          $x_{ir_2}^{t+1} = x_{r_2}^t$;
          $x_i^{t+1} = x_i^t - w \times (x_{ir_2}^t - x_{best}^t) \times 2 \times (rand - 1)$ ;
        else
          $x_i^{t+1} = x_i^t - w \times (x_{ir_1}^t - x_{best}^t) \times 2 \times (rand - 1)$ ;
        end if
      else
        $x_i^{t+1} = x_{min}^t + rand \times (x_{max}^t - x_{min}^t)$;
      end if
    end if
  end for
Update T if there is a better solution;
$iter = iter + 1$;
end while
Return the best solution of T;
```

Algorithm 8 The Pseudocode *of EGOAMLP*

```
1: Initialize all the parameters such as:
     Training parameters;
     Maximum No. of iterations ($iter_{max}$), $c_{max}$, $c_{min}$, and number of population ($N$);
     Probability (P) of applying EGOA operator on ANN structure or Weights & Biases;
2: Generate a random population ($X_i^d$) :
     ($i = 1, 2, 3 \ldots, N$) and ($d = 1, 2, \ldots.$ dim ➜ no. of dimensions);
3: Calculate the fitness of each grasshopper;
4: for each grasshopper do
5:     Calculate the MSE for the grasshopper by Eq. (25);
6:     if the current MSE < the global minimal MSE then
7:         Update the global minimal MSE
8:     end if
9: end for
10: $\hat{T}_d$ = the best grasshopper;
11: While ($iter < iter_{max}$)
12:     Update the parameter c using Eq. (8);
13:     If (rand < P)
14:         Apply EGOA on structure of the solution and apply equation 17 on the final result from EGOA.
15:         Add or remove the random nodes in weights & biases
16:     else
17:         Apply EGOA on weights & biases of the solution
18:         Build the structure from the parent grasshoppers
19:     end if
20:     for each grasshopper do
21:         Calculate the MSE for the grasshopper by Eq. (25);
22:         if the current MSE < the global minimal MSE then
23:             Update the global minimal MSE
24:         end if
25:     end for
26:     Update T if there is a better solution;
27:     Save the current best solution with the minimal MSE;
28:     $iter = iter + 1$;
29: end while
30: Return the best solution of the minimal MSE;
```

There are three fundamental methods applied to train NNs by using metaheuristic algorithms. Firstly, the metaheuristic is utilized to find a suitable ANN structure for the NN during the learning process, where the training algorithm determines the best architecture of the NN model for solving a particular problem. Changing the architecture can be accomplished by manipulating the connections between the neurons, the number of hidden layers and the number of hidden neurons in each layer. Secondly, metaheuristics are used to find a combination of weight and bias that provides a minimum MSE which represents the cost function of the NN training. The training algorithm searches for suitable values for all connection weights and biases to minimize the overall error of the ANN. Lastly, the metaheuristic is used to adjust the parameters of the gradient descent learning algorithms. The previous studies such as [64], [66], [67] have been applied based on the second method which is finding the optimal weights and biases during the training process. However, the present study applies the GOA algorithm that was proposed recently, to find the optimal ANN architecture and optimize its weights and biases.

The general training pseudo-code of the EGOA is shown in Algorithm 8 and is also depicted in Figure 3. In line 1, all the parameters of the EGOA algorithm and the NN model are initialized, namely, c_{max}, c_{min}, $iter_{max}$, and the lower and upper bounds. In line 2, a set of solutions is generated randomly. The EGOA algorithm has many variables, including solution vector size, which represents the number of solutions in the solution vector (SV). In the SV solutions, every solution $x_i(i = 1, 2 \ldots, D)$ is a D-dimensional vector, where the dimension of the solutions is defined by Eq. (16). The D is the number of decision variables. The ranges of lower and upper limits are specified by two vectors x_L and x_U, where both having the same length of the solution vector. The SV is a vector of the best solution vectors achieved so far. It is an augmented vector of size SV × D, which is composed as indicated in Eq. (16). The SV size is set prior to run the algorithm. Each solution vector is also associated with a quality value (fitness) based on the objective function $f(x)$. Figure 3 shows that EGOAMLP algorithm is similar to other optimization algorithms, where it begins by initializing the random solution memory vector representing candidate MLP weight values. Line 3 calculates the initial fitness value for each grasshopper (solution). In lines 4-9, the MSE for each grasshopper (solution) in the whole SV is checked and the MSE of the global minimum is derived. In line 10, the EGOA parameter, namely, $\hat{T}_d$ is calculated. In line 11, all the loops are assigned to maximum iteration. In line 12, coefficient c of the EGOA is updated.

$$SV = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ x_{31} & x_{32} & \cdots & x_{3D} \\ \cdots & \cdots & \ddots & \cdots \\ x_{SV1} & x_{SV2} & \cdots & x_{SVD} \end{bmatrix} \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{SV}) \end{bmatrix} \tag{16}$$

In line 13-19, a little variation is introduced to the original EGOA operator by first testing a probability parameter P and selecting one of the two possible paths to balance the application of the EGOA operator to either the ANN structure or its weights/biases. This parameter is set to 0.5 in order to give both cases the same opportunity to improve the quality of solution.

In each generation, the opportunity to identify the solution as a parent is proportional to the quantity by which its fitness is less than other of the other solution's fitness. The optimization process of the EGOA algorithm is applied with the probabilities of 50% on the ANN structure of the parents, and there are 50% chances that the optimization process would be applied to weights and biases of the parents during the current iteration. If the optimization process of the EGOA algorithm is to be applied to the structure of parents, then the weights and biases of neurons are randomly added or removed to fit the ANN structure of the solution. The number of neurons in weights and biases solution vector is calculated based on equations (18, 19). For example, if the size of the new child solution is smaller than its parents' solution, then based on a new ANN structure a number of neurons are randomly removed from the weights and biases solution vector. If the child solution size is greater than its parents' solution, then a number of neurons are added into the weights and biases solution. However, if the optimization process by the EGOA algorithm is to be applied to weights and biases, then the ANN structure of the child solution is inherited from its parents' solution.

This mechanism allows the EGOAMLP to have an extensive variety of solutions with the optimal ANN architectures and optimizes values of weights and biases. An effective fitness function that considers both the number of the connections (weights and biases) and the error to be minimized helps the algorithm to optimize the fitness function with a small size model. The population of the EGOAs algorithm is updated based on new solutions. The iterations are stopped when the number of iterations exceeds the maximum number of generations. Furthermore, this method needs to check which connection weight and bias are active and which one is inactive to find the architecture of the ANN. The selection of the architecture of the NN is the responsibility of the EGOA algorithm when a new solution is generated. The pseudocode and the schematic procedure of EGOAMLP are shown in Algorithm 2 and Figure 3, respectively. The solutions in the population consist of two parts. The first part encodes the architecture of the ANN. When the result of applying the random operator, P decides that the structure of the MLP is to be adjusted by the new position of the grasshopper, the search space is formed as a binary pattern [0, 1] to represent the grasshopper's position in the binary vector using a sigmoid function as shown in Eq. (17).

$$f(x) = {}^{1}\!/_{1+e^{-x}} \tag{17}$$

Eq. (17) is applied on the output of Eq. (7) when the optimization process of the EGOA algorithm is applied to the structure of the ANN. If the result of the Eq. (17) is less than a certain value within the range of (0, 1) then the result of Eq. (17) is set to zero, otherwise, this result is adjusted to one.

The second part determines that the weights and biases in the ANN model are to be adjusted. The movement of each grasshopper in the search space is towards continuous-valued places between [−1, 1]. For the initial population, the structure of the solution is randomly assigned, and then the length of weights and biases is calculated to fit each structure. In the end, the values of the weights and biases are randomly generated. In lines 20-25, the MSE for each grasshopper (solution) in the whole solution vector is checked and the MSE of the global minimum is derived. In line 26, the EGOA parameter, namely, $\widehat{T_d}$ is updated. Line 27 saves the current best solution with the minimal MSE. In line 28, the *iter* parameter is increased by 1. Finally, in line 30, the best solution to the minimal MSE is identified.

1) SOLUTION REPRESENTATION OF THE STRUCTURE AND WEIGHTS/BIASES FOR ANN USING GOA-MLPT

In this part, shows the details of the EGOAMLP model to improve the MLP architecture and optimizes its weights and biases for the purpose of classifying email. The goal behind the EGOAMLP model is to reduce the overall error of accuracy. Furthermore, any MLP model relies on the number of hidden layers, number of neurons in each hidden layer, connection weights, biases for each layer and the number of inputs. The biases relate to each neuron in the hidden and output layers. In the EGOAMLP model, the solution is represented by two one-dimensional vectors:

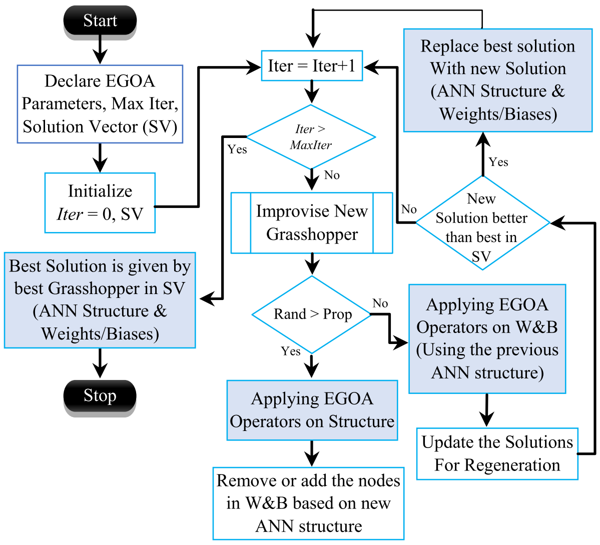
Start
Declare EGOA Parameters, Max Iter, Solution Vector (SV)
Initialize Iter = 0, SV
Best Solution is given by best Grasshopper in SV (ANN Structure & Weights/Biases)
Stop
Iter = Iter+1
Iter > MaxIter
Yes
No
Improvise New Grasshopper
Rand > Prop
Yes
No
Applying EGOA Operators on Structure
Remove or add the nodes in W&B based on new ANN structure
Replace best solution With new Solution (ANN Structure & Weights/Biases)
New Solution better than best in SV
Applying EGOA Operators on W&B (Using the previous ANN structure)
Update the Solutions For Regeneration

FIGURE 3. Adapting EGOA Algorithm Solution Vectors as ANN Structure and Weights/Biases.

1. ANN structure solution vector represents the number of inputs, the number of hidden layers and the number of neurons in each hidden layer in the ANN model.
2. Weights and biases solution vector represents the weights and biases in the trained MLP model.

Each of these two solution vectors has a different representation as shown in Figure 4. The value in the structure solution vector contains 0 or 1, whilst each value in the weights and biases solution vector contains a real number in the range of $[-1, +1]$. The objective here is to develop a training algorithm that excels in finding optimal values for all connection weights and biases and the structure of the NN, which will ultimately contribute to minimizing the overall MLP classification error. The structure solution vector of the EGOAMLP model is divided into three parts. The first part contains a set of cells representing the neurons in the input layer, which are considered the features of the datasets; the second part contains three cells representing the number of the hidden layers; the third part contains three cells that are used for the number of nodes in each hidden layer. Each number in these cells is represented as a binary string. For example, Figure 4 shows the binary string "010" in the second part of the structure solution vector, which indicates that there are two hidden layers in the MLP model. The representation in the first part, corresponding to the input neurons, is a little different. The bits in the binary string of this part refers to the individual input features in each sample. Therefore, the number of cells in this part is equal to the number of features in the input dataset, and a value of "1" indicates the existence of that particular feature in the current sample, while a value of "0" indicates that the feature does not exist in the input sample. All the experiments in this study use the whole set of features in the dataset.

The length of the weights and biases solution vector is equal to the number of weights in each layer of the MLP model, in addition to the number of biases in each layer. This length is computed using Eq. (18). As such the total number of weights and biases depend on the total number of hidden layers as well as the number of nodes per layer, as shown in Equations (19) and (20).

$$\text{Length of weights and biases vector} = W + B \tag{18}$$
$$W = (I \times N) + ((N \times N) \times (H - 1)) + (N \times O) \tag{19}$$
$$B = H \times N + O \tag{20}$$

where W is the number of weights, B is the number of biases, I is the number of nodes in the input layer, N is the number of nodes in each hidden layers, H is the number of hidden layers, and O is the number of nodes in the output layer. With regard to identifying the number of hidden nodes in the MLP network, there are several rules proposed in the state-of-arts and there is no agreement among researchers on the best rule of application. However, in initialization step of this work selects a common rule that has been adopted by many previous works such as those in [40], [66].

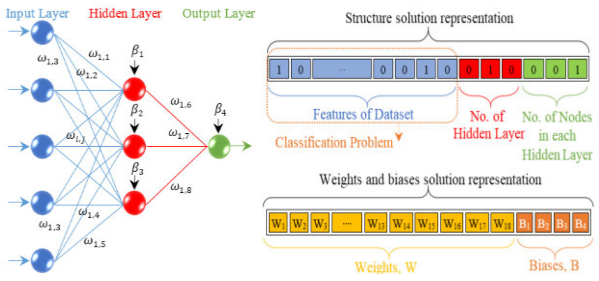

FIGURE 4. The solution representation of EGOAMLP algorithm.

2) ADAPTING THE GRASSHOPPER QUALITY MEASURE (FITNESS FUNCTION)

In this part, to determine how good the improvised solution is, the GOA algorithm uses a grasshopper quality measure, i.e., fitness function. The fitness function is used to assess the quality of the solutions in successive generations. With the use of the fitness function, a solution is selected that optimizes the quality of the solution. This fitness function plays the role of objective functions in optimization algorithms. The goal here is to minimize the values obtained by this fitness function. This objective is similar to the aim of the training methods in previous studies [40], [66], which is to reduce the overall error. Consequently, the fitness function can utilize any of the ANNs error calculation formulas or develop a new measure based on these formulas.

In this study, the fitness function, $f(s)$ of a solution is computed by the MSE, which is the main quality measure for the proposed EGOA training algorithm. The goal of the training algorithm is to minimize the MSE up to reaching the maximum number of iterations. The MSE is one of the most used fitness functions. Because this study focuses on classification problems, the MSE, as the main fitness function, measures the quality of solution vectors, sorted from best to

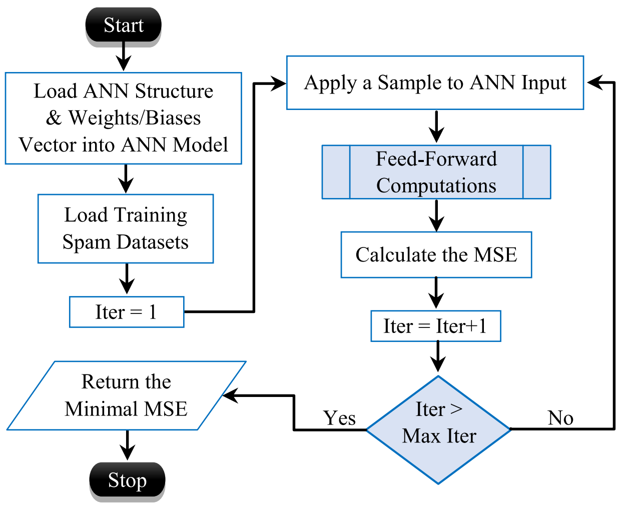
Start
Load ANN Structure & Weights/Biases Vector into ANN Model
Load Training Spam Datasets
Iter = 1
Apply a Sample to ANN Input
Feed-Forward Computations
Calculate the MSE
Iter = Iter+1
Iter > Max Iter
Yes
No
Return the Minimal MSE
Stop

FIGURE 5. MLP feed-forward computations to compute MSE.

worst with the best being the lowest value of MSE. Thus, to find an optimal solution, i.e., the MLP with the best weights and biases vector, the MSE value must be the smallest among the ones in the current solution vector.

Firstly, the feed-forward computations must be performed in order to compute MSE on the given MLP structure. This is a repetitive process that involves the loading of the entire training dataset. This would require a process by which the network weights and biases, represented by the solution vector, are to be loaded into the MLP structure to implement such a computation. The MLP structure must be therefore flexible to allow loading of different weight and bias vectors during the EGOAMLP algorithm initialization and update processes. The feed-forward computation process is shown in Figure 5.

The objective of training the MLP is to achieve the highest classification, approximation, or prediction accuracy for both training and testing samples. In this work, a similar methodology used by several studies [68] was applied to calculate the fitness function. Figure 5 illustrates that the MLP has three layers including input, hidden, and output layer. Assuming the number of input nodes is N, the number of hidden nodes is H, and the number of output nodes is O, then the output of the i^{th} hidden node is calculated as follows:

$$\begin{aligned} f\left(S_j\right) &= Sigmoid\left(S_j\right) \\ &= 1/\left(1+exp\left(-\left(\sum\nolimits_{i=1}^{N} \mathcal{W}_{ij}.\mathcal{X}_i-\beta_j\right)\right)\right), \\ &\quad j=1,2,\ldots,H \end{aligned} \tag{21}$$

where $S_j = \sum\nolimits_{i=1}^{N} \mathcal{W}_{ij}.\mathcal{X}_i - \beta_j$, $\mathcal{W}_{ij}$ is the connection weight from the i^{th} node in the input layer to the j^{th} node in the hidden layer, β_j is the bias (threshold) of the j^{th} hidden node, and $\mathcal{X}_i$ is the i^{th} input. After calculating the outputs of the hidden nodes, the final output can be defined as follows:

$$\mathcal{O}_k = \sum\nolimits_{i=1}^{N} \mathcal{W}_{kj}.f\left(S_j\right) - \beta_k, \quad k=1,2,\ldots,O, \tag{22}$$

where W_{kj} is the connection weight from the j^{th} hidden node to the k^{th} output node and β_k is the bias (threshold) of the k^{th} output node. Finally, the learning error E (fitness function) is calculated as follows:

$$E_k = \sum\nolimits_{i=1}^{O} \left(\mathcal{O}_i^k - d_i^k\right)^2 \tag{23}$$

$$\mathrm{MSE} = \sum\nolimits_{k=1}^{q} \frac{\mathrm{E_k}}{\mathrm{q}} \tag{24}$$

where q is the number of training samples, d_i^k is the desired output of the i^{th} input unit when the k^{th} training sample is used, and $\mathcal{O}_i^k$ is the actual output of the i^{th} input unit when the k^{th} training sample is used. Therefore, the fitness function of the i^{th} training sample can be defined as follows

$$\mathrm{Fitness}(x_i) = \mathrm{MSE}(x_\mathrm{i}) \tag{25}$$

IV. PERFORMANCE EVALUATION AND DISCUSSION

All the fair experiments for compared models were conducted using a laptop loaded with Core i5 2.4 GHz CPU and 8 GB RAM, and MATLAB R2014a running on a Windows 7, and no commercial GOA based software was utilized in this study. To evaluate the performance of the developed model, we used three datasets composed of sets of email spam messages. The sets of messages are marked as spam or non-spam. These datasets are commonly used in evaluating spam detection systems. The SpamBase dataset is one of the prominently ancient datasets, which is still being used in the current research. The SpamAssassin dataset is a free, open-source, flexible, and powerful spam-fighting tool where it is characterized by removing duplicate and irrelevant data. Other datasets have also emerged in recent years for evaluating SDSs, such as the UK-2011 Webspam dataset, which is developed in 2011.

To evaluate the performance of the EGOAMLP-SDS model developed in this study, each dataset was divided into two parts: one subset is used in the training phase so that the NN can learn from the data (therefore, these data are labeled), and another subset to test the trained NN on new data that have not been seen during the training (therefore, these samples are not labeled), as illustrated in Figure 1. The dataset can be divided into two percentages during the evaluation process, among which 50/50, 70/30 and 60/40 ratios for training and testing datasets, respectively, are common in the literature. In this work, all data sets are randomly split into 70% for training and 30% for testing.

A. DATASET PREPROCESSING

The most important step in this study is processing the datasets before using them. The dataset pre-processing is divided into two stages as showed in Figure 1. In the first stage, the dataset used to evaluate the algorithm is determined. As the size of this dataset is too huge to load into memory, random sample records from the dataset are selected. This random sample is divided into two subsets, the first is known as the training dataset, and the second is known as the testing dataset. In the second stage of the pre-processing, all symbols and characters are converted to a numeric value if they are not numeric. These numeric values are then normalized by using the normalization Eq. 26. To convert the raw email formats into numeric values, such as the case with the SpamAssassin dataset, a feature extraction tool is used, which is available at this link ''https://github.com/7ossam81/Email Features Extraction''.

The main objective of normalization is to make numeric values of different features in the same range. All the features of the dataset must be normalized before applying it in the training and testing processes. It is intended to produce regular semantics of the feature values in the datasets. The normalization process puts all features on the same scale by converting the values into the range [0, 1] using Eq. (26) as follows:

$$x_{new} = \frac{x_{current} - x_{min}}{x_{max} - x_{min}} \tag{26}$$

All the three datasets used in this study have a class for each set of features, in which the class is either not spam or spam email. Thus, each record in the dataset belongs to one of the two classes, non-spam or spam. The value of each class is mapped to a numeric value, specifically, the non-spam email class is mapped to the number 0 and the spam email class is mapped to 1.

1) SPAMBASE DATASET

The SpamBase dataset benchmark is widely popular in the process of evaluating spam detection systems. This dataset is consists of 4601 messages (instances), each of which comprises 57 features. Approximately 1813 (39%) of the messages are marked as spam and 2788 (61%) are identified as non-spam. The dataset was acquired from the University of California at Irvine (UCI) ML Repository [69]. The collection of features in this dataset has been based on the frequency of some selected words and special characters in the e-mails, while the not spam message was contributed by Forman; this was obtained from a single mailbox.

TABLE 1. All features of the spambase dataset.

No. of Feature	Feature type	Feature description
48	word_freq_WORD	Word frequency expressed as a percentage
6	char_freq_CHAR	Char frequency expressed as a percentage
1	capital_run_length_average	Average length of uninterrupted sequences of capital letters
1	capital_run_length_longest	Average length of uninterrupted sequences of capital letters
1	capital_run_length_total	Total number of capital letters in the e-mail
1	Class feature	Class feature {0 = Not spam, 1 = Spam}

This dataset is already pre-processed, though most other datasets come in their raw format. Among the 57 features, 48 are represented by words generated from the original messages with the absence of stop-list or stemming, and they are considered and enlisted as the most unbalanced words for the class spam. Table 1 shows all the features in this dataset. The remaining six features are the percentage of appearance of the special characters ";", "(", "[", "!", "$", and "#". The other three features are a representation of different measures of the appearance of capital letters that exist in the text of the messages. Finally, the class label of each instance can be 0 for non-spam or 1 for spam. SpamBase dataset is one of the best datasets that performs well during learning and evaluation techniques.

2) SPAMASSASSIN DATASET

SpamAssassin is a widely used email spam dataset that is well-known for spam filters. This dataset is currently publicly available on Kaggle under the name ''SpamAssassin public corpus'' [70]. Kaggle is a website that is a transparent repository for public datasets and this dataset was provided to the website by the UCL ML repository and other public open datasets. The SpamAssassin public corpus was publicly released in 2005. It is made up of sets of private emails contributed by individual users or collected from the public. Unlike the SpamBase dataset, all the email headers are produced in full in SpamAssassin. The SpamAssassin dataset is considered one of the important datasets used to evaluate the performance of spam detection systems. This dataset contains a total of 6047 legitimate (ham) and unsolicited (spam) emails with 1,897 spam emails (31.4% spam ratio), 3,900 easy non-spam emails, which does not contain spam signatures (such as HTML, HTML mark-up, colored text, spammish-sounding phrases, etc.), and 250 hard but legitimate emails which are closer to typical spam in many respects (not shown in Table 2). This unusual distribution of the email types makes the data imbalanced and for that reason more challenging. In this research work, words are taken as being separated by white space. The SpamAssassin corpus uses individual characters such as '$' and '#', even though the email may contain web links, HTML scripts, pictures, attachments, etc. This study will focus on the text content of the messages.

TABLE 2. Analysis of spamassassin email messages.

Type	Message Number	Describe
Spam	500	Received from non-spam trap sources
Easy_Ham	2500	Non-spam messages. Easy to differentiate from spam. Do not contain any spam signature (E.g., HTML)
Easy_Ham_2	1400	Non-spam messages
Spam_2	1397	Spam Messages
Total	5797	33% Spam ratio

3) UK-2011 WEBSPAM DATASET

The UK-2011 Webspam dataset consists of 3766 instances with 11 features. Each example in the data is labeled as Ham or Spam. This dataset was built by using part of the UK-2007 Webspam dataset, to create an Extended-UK-2011 Webspam dataset and retrieving a new part of UK spam web pages for evaluation purposes. It includes 1768 ham emails and 1998 spam emails. The percentage of spam emails form approximately 53% of the emails, which makes the data imbalance and therefore more challenging. Table 3 shows all the features in this dataset, and the full description of the features can be found in [71], [72].

TABLE 3. Feature descriptions of the UK-2011 webspam dataset.

No. of Feature	Feature Description
1	Existing amount of visible and clickable text in a hyperlink (i.e., anchor text)
2	Number of words
3	Average word length
4	Number of words in the title elements (because spammers tend to use unrelated characters to enhance the page rank
5	Page compression rate
6	Number of unique words
7	Number of characters in the meta-element (because spammers tend to utilize keyword stuffing to enhance the page rank)
8	Number of words in the meta element (because spammers tend to utilize keyword stuffing to enhance the page rank)
9	Longest word (because spammers tend to utilize long words to increase the page rank)
10	Shortest word (because spammers tend to utilize long words to increase the page rank)
11	Number of images

4) TEST OPTIMIZATION FUNCTION

To obtain reliable comparison, common control parameters of all the algorithms have been set as the same values. Detail of all algorithm's parameters shown in Table 4.

B. ALGORITHMS AND PARAMETERS

In this study, various algorithms have been studied to ensure a reliable analysis of the performance of the proposed model. To ensure a reliable comparison, all the common control parameters of all algorithms were set to the same values, including the solution vector size SV, and the dimensionality of the search space D that represents the number of features of the dataset. The parameters of all algorithms used in this study are presented in Table 5.

C. EVALUATION CRITERIA FOR SDS

In this study, the proposed model has been compared and evaluated based on the Accuracy (ACC), Detection Rate (DR) and False Alarm Rate (FAR), which are widely used in literature to evaluate the performance of SDSs. These metrics include the True Negative (TN), True Positive (TP), False Negative (FN), and False Positive (FP) rates. These four main criteria are collected from the confusion matrix, which is a commonly used tool to describe the performance of NN classifiers. Table 6 shows the confusion matrix for a two-class classification. The performance metrics are shown in Table 7.

The abbreviations of the confusion matrix for a 2-class classification are:

- (TP): indicates that an email classified as spam email is actually spam email.
- (TN): indicates that an email classified as normal email is actually non-spam email.
- (FP): represents a non-spam email that is classified as a spam email.
- (FN): represents a spam email that is classified as a non-spam email.

TABLE 4. Test optimization functions.

No.	*Name*	*Equation*
1	*Sphere*	$f(x) = \sum_{i=1}^{n} x_i^2$
2	*Schwefel 2.22*	$f(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$
3	*Schwefel 1.2*	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$
4	*Schwefel 2.21*	$f(x) = max_i\{\lvert x_i\rvert, 1 \le i \le n\}$
5	*Rosenbrock*	$f(x) = \sum_{i=1}^{n-1} \left[100\sqrt{\lvert x_i - x_i^2\rvert} + (1 - x_i)^2\right]$
6	*Step*	$f(x) = \sum_{i=1}^{n} \lfloor\lvert x_i\rvert\rfloor$
7	*Quartic*	$f(x) = \sum_{i=1}^{n} ix_i^4 + rand[0,1)$
8	*Schwefel 2.26*	$f(x) = -418.983 \sum_{i=1}^{n} \left[x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)\right]$
9	*Rastrigin*	$f(x) = \sum_{i=1}^{n} [x_i^2 - 10 \cos 2\pi x_i + 10]$
10	*Ackley*	$f(x) = -20e^{\left(-0.2\times\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)} - e^{\left[\frac{1}{n}\sum_{i=1}^{n} \cos 2\pi x_i\right]} + 20 + e^1$
11	*Griewank*	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
12	*Penalized No.1*	$f(x) = \frac{\pi}{n} \times \{10sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10sin^2(\pi y_{i+1})]$ $+(y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10,100,4)$
13	*Penalized No.2*	$f(x) = 0.1 \times \{sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + sin^2(3\pi x_{i+1})]$ $+(x_n - 1)^2[1 + sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5,100,4)$
14	*Shekel's foxholes*	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{i,j})^6}\right]^{-1}$
15	*Kowalik's*	$f(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_{1(b_i^2 + b_i x_2)}}{b_i^2 + b_i x_3 + x_4}\right]^2$
16	*Six-hump Camel-back*	$f(x) = 4x_2^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$
17	*Branin*	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$
18	*Goldstein Price*	$f(x) = [1 + (x_1 + x_2 + 1)^2$ $\times (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $\times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$
19	*Hartman's*	$f(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2\right]$, *n=3*
20	*Hartman's*	$f(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2\right]$, *n=6*
21	*Shekel's Family*	$f(x) = -\sum_{i=1}^{m}[(x - a_i)(x - a_i)^T + c_i]^{-1}$, *m=5*
22	*Shekel's Family*	$f(x) = -\sum_{i=1}^{m}[(x - a_i)(x - a_i)^T + c_i]^{-1}$, *m=7*
23	*Shekel's Family*	$f(x) = -\sum_{i=1}^{m}[(x - a_i)(x - a_i)^T + c_i]^{-1}$, *m=10*
24	*Sum squares*	$f(x) = \sum_{i=1}^{n} ix_i^2$
25	*Dixon-price*	$f(x) = (x_1 - 1)^2 + \sum_{i=1}^{n} i(2x_i^2 - x_{i-1})^2$
26	*Levy*	$f(x) = sin^2(\pi w_1) + \sum_{i=1}^{n-1} (w_i - 1)$ $[1 + 10sin^2(\pi w_i + 1)] + (w_n - 1)^2[1 + sin^2(2\pi w_n)]$
27	*Trid*	$f(x) = \sum_{i=1}^{n}(x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$
28	*Powell*	$f(x) = \sum_{i=1}^{n/4} \left[\begin{matrix}(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} + x_{4i})^2 \\ +(x_{4i-2} + 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4\end{matrix}\right]$
29	*Michalewicz*	$f(x) = -\sum_{i=1}^{n} sin(x_i)\left(sin(ix_i^2/\pi)\right)^{2m}$
30	*Zakharov*	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$
31	*Bent cigar*	$f(x) = X_1^2 + 10^6 \sum_{j=1}^{i} x_j^2$

TABLE 4. ***(Continued.)*** **Test optimization functions.**

32	*Styblinski -Tang*	$f(x)=\frac{1}{2}\sum_{i=1}^{n}(x_i^4-16x_i^2+5x_i)$
33	*Sum of different powers*	$f(x)=\sum_{i=1}^{n}\lvert x_i\rvert^{i+1}$
34	*Rotated Hyper-ellipsoid*	$f(x)=\sum_{i=1}^{n}\sum_{j=1}^{i}x_j^2$

TABLE 5. **The parameters of the models used in the performance analysis.**

Alg.	Parameter	Symbol / Abbr.	Value
ABC	*limit*	*limit*	100
ALO	*linear decreased*	*a*	2
	random walk	*t*	[0, 1]
BBO	*mutation probability*	*P*	*0.01*
	elitism parameter	*keep*	*2*
CS	*rate of alien eggs/solutions*	pa	0.25
DA	-	*r1,r2*	*[0, 1]*
	-	β	*1.5*
	separation weight	*s*	*0.1*
	inertia weight	*w*	*0.2-0.9*
	alignment weight	*a*	*0.1*
	cohesion weight	*c*	*0.7*
	food factor	*f*	*1*
	enemy factor	*e*	*1*
DE	*weighting factor*	*F*	0.5
	crossover constant	c_r	0.5
GOA	-	C_{min}	0.00004
	-	C_{max}	1
	number of search agents	-	5
GSA	-	g_0	100
	-	$\propto$	0.2
HS	*harmony memory size*	*HMS*	50
	harmony memory consideration rate	*HMCR*	0.95
	Pitch adjustment rate	*PAR*	0.1
MBO	*butterfly adjusting rate*	*BAR*	0.4167
	max step	*Smax*	1.0
	migration period	*peri*	1.2
	migration ratio	*p*	0.4
MFO	*linearly decreased*	*r*	-1 to -2
	logarithmic spiral	*b*	2
	random number	*t*	[-1, 1]
PBIL	*habitat modification probability*	-	1
	immigration probability	-	[0, 1]
	step size for numerical integration	-	1
	maximum immigration	-	1
	mutation probability	-	0.005
PSO	*inertial constant*	-	0.3
	cognitive constant	-	1
	Social constant	-	1
SCA	*random number*	r_1, r_2, r_3, r_4	[0, 1]
	linear decreased	*a*	2
WOA	*linearly decreased*	*a*	*2 to 0*
	random vector	$\vec{r}$	[0, 1]
	coefficient vectors	$\vec{A}$	[-1, 1]
	coefficient vectors	$\vec{c}$	[1, 1]
	random number	*l*	[-1, 1]
	random number	*p*	[0, 1]

V. RESULTS AND DISCUSSION

In this section, the evaluated the performance of the proposed algorithms, the tests were performed in two parts: In the first part, the tests were performed on the benchmark functions using the proposed EGOAs approaches and comparing the performance of the proposed algorithms with the performance of the standard GOA, old optimization algorithms, and a new optimization algorithm for the purpose a total of 34 low and high-dimensional benchmark functions were used. The population size was set to 50 in all scenarios and the same for the maximum number of generations. We also report the results for every single scenario based on 30 runs to reduce the impact of chance on individual runs. These tests will be examined in detail in Scenarios (1, 2, and 3). The second part, focused on testing the performance of the proposed approach EGOAs, in the detection of spam emails. These tests will be examined in detail in Scenarios (4, 5, and 6).

TABLE 6. **The confusion matrix for a 2-class classification.**

Predicted \ Actual	Non-Spam	Spam	Total
Non-Spam	TP	FP	TP+FP
Spam	FN	TN	FN+TN
Total	TP+FN	FP+TN	

TABLE 7. **Performance matrix for classification.**

Measure	Definition	
Accuracy (ACC)	$ACC=\frac{TP+TN}{TP+TN+FP+FN}$	(27)
False alarm rate (FAR)	$FAR=\frac{FP}{FP+TN}$	(28)
Detection rate (DR)	$DR=\frac{TP}{TP+FN}$	(29)
Sensitivity (SN)	$SN=\frac{TP}{TP+FN}$	(30)
Specificity (SP)	$SP=\frac{TN}{TN+FP}$	(31)
Positive predictive value (PPV)	$PPV=\frac{TP}{TP+FP}$	(32)
Negative predictive value (NPV)	$NPV=\frac{TN}{TN+FN}$	(33)
F-Measure (F1)	$F1=\frac{2\times PPV\times SN}{PPV+SN}$	(34)
Matthews correlation coefficient (MCC)	$MCC=\frac{(TP\times TN-FP\times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN}}$	(35)
G-mean (GM)	$GM=\sqrt{SN\ \times SP}$	(36)

A. SCENARIO 1: PERFORMANCE OF EGOA AGAINST THE STANDARD GOA ALGORITHM

The experimental test of scenario 1 verified the effectiveness of the six proposal for an enhanced GOA against the original GOA. The results of comparing the six proposed algorithms and original GOA algorithm to solve 34 global numerical optimization problems are given in Table 8 below. Due to a large number of optimization functions, the results (best values) were presented the benchmarks are divided into 2 types that can evaluate the different capabilities of the algorithms: Benchmarks F14-F23 of the Low-Dimensional Functions (LDF) that are set to (2, 3, 4, and 6) dimensions, and benchmarks F1-F13 and F24-F34 of the High-Dimensional

Functions (HDF) that are set to (10, 30, 60, and 90) dimensions, listed in Table 4. The results of this table represent the best results achieved by each algorithm at all. The "best result" here means the closest result to the actual optimal value of the function, as per Table 4. The results for each benchmark function include a group of lines (row) each corresponds to a different dimension of the function as shown in the table. This table introduces the result of the minimum value (closest value to the global optimum) achieved by each algorithm after the 50 generations with run repeated 30 times. To highlight the best performance, we distinguish the best minimum value (best result) for each benchmark function in bold.

Table 8 presents the best value results for 10 functions out of 34 functions. It is evident from the table that the six proposed algorithms give the best optimal results on all functions low dimensional benchmark functions of (2, 3, 4, and 6), with the exception of the E3GOA algorithm shows the best result on function (F15) in dimension 4 by comparing it with the six proposed algorithms. The seven compared algorithms show similar results on F16, F17, F18, F22, and F23, especially on lower dimensioned functions. The six proposed algorithms the best result on function (F14) in dimension 2 by comparing it with the GOA algorithm. From all 10 experiments in Table 8, the GOA algorithm was beaten only five times in terms of the best optimum.

Table 8 clearly shows that the six proposed algorithms give the best optimal results on all functions especially on dimensions of 10 and above, with the exception of four functions, F8, F27, F29, and F32. From all 106 experiments in Table 8 (10 functions $\times$ 1 dimension for each LDF, 24 functions $\times$ 4 dimensions for each HDF), the six proposed algorithms were beaten only eight times in terms of the best optimum value. The GOA algorithm shows the better best value on one function (F27) in dimension 10, on tow functions (F29, and F32) in dimension 30, on two functions (F8, and F29) in dimension 60, and on two functions (F29, and F32) in dimension 90.

To analyze the performance of the algorithms in terms of their convergence speed, we plot the progression of each algorithm in a single run over all 50 iterations, showing the best optimum found by the algorithm per iteration. Due to the limited space, 4 results out of 106 comparisons are shown to highlight the most convergent curves of the six proposed algorithms against the original GOA. Figures 6(a–d) show a sample of such convergence plots for 4 benchmark functions.

Figure 6(a) shows the results of the seven algorithms when applied to Zakharov function with 10 dimension. The result for Zakharov function in Figure 6(a) shows that the six proposed algorithms outperform the original GOA algorithm again. Although the rate of convergence in the case of GOA and the six proposed algorithms are similar in Figure 6(a), there is a significant difference in the result with our proposals. Figures 6(b, c) show the results against the Schwefel 2.21, and the Ackley functions respectively with 30 dimensions. The six proposed algorithms show an

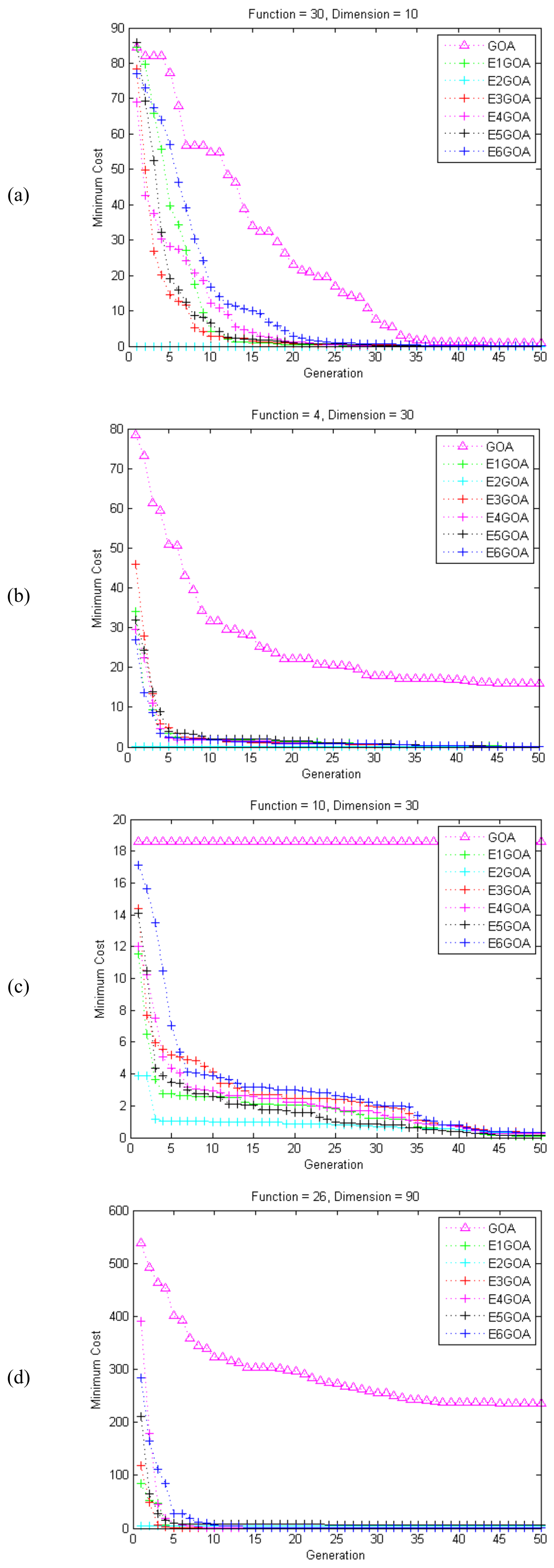

FIGURE 6. Convergence curves of GOA, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, AND E6GOA against 4 functions (A-F30, B- F4, C-F10, D-F26) with 10, 30, and 90 dimensions over 50 generations.

TABLE 8. The best results obtained by the E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, E6GOA and GOA on the test optimization functions in (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.

F	Dimension	GOA	E1GOA	E2GOA	E3GOA	E4GOA	E5GOA	E6GOA
F1	10	7.1E-02	3.0E-04	**0.0E+00**	1.1E-02	2.9E-04	1.3E-04	5.2E-03
	30	1.2E+02	1.3E-08	**0.0E+00**	1.9E-05	1.6E-01	2.3E-06	2.2E-02
	60	2.5E+03	1.0E-03	**5.8E-06**	5.1E-04	5.3E-02	3.3E-02	8.4E-05
	90	1.1E+04	1.8E-05	**0.0E+00**	3.8E-05	2.8E-04	2.1E-06	3.8E-02
F2	10	5.7E-01	2.4E-04	**0.0E+00**	3.4E-03	6.4E-03	2.3E-03	8.7E-03
	30	1.1E+02	2.1E-03	**0.0E+00**	1.4E-02	2.2E-02	2.9E-03	1.5E-03
	60	2.5E+02	2.0E-03	**0.0E+00**	2.8E-02	8.4E-02	3.2E-02	6.9E-02
	90	4.0E+02	2.1E-03	**0.0E+00**	7.1E-02	1.2E-02	1.1E-02	1.1E-02
F3	10	9.1E+01	2.2E-07	**0.0E+00**	3.1E-02	9.7E-02	1.8E-02	2.6E-03
	30	5.5E+03	9.2E-05	**0.0E+00**	2.9E-01	8.2E-03	2.5E-01	5.6E-01
	60	5.1E+03	8.6E-05	**0.0E+00**	2.9E+00	2.4E-02	2.6E-02	1.9E-01
	90	4.9E+04	1.0E-04	**0.0E+00**	4.9E-01	1.0E+01	3.2E+00	2.2E+00
F4	10	4.0E-01	1.2E-02	**0.0E+00**	1.3E-02	4.9E-03	7.7E-03	6.0E-03
	30	1.6E+01	2.1E-03	**0.0E+00**	8.5E-03	1.1E-02	3.1E-03	1.6E-03
	60	2.1E+01	7.1E-04	**0.0E+00**	3.7E-02	2.4E-04	1.0E-02	1.6E-02
	90	2.4E+01	1.6E-03	**0.0E+00**	8.7E-03	1.0E-03	2.3E-02	4.3E-02
F5	10	4.5E+02	3.9E-02	8.3E+00	7.9E-03	1.9E-03	8.9E+00	**1.3E-03**
	30	2.3E+03	9.0E-02	2.9E+01	**2.6E-05**	2.3E-03	2.9E+01	5.5E-04
	60	5.7E+05	3.4E-02	5.9E+01	6.8E-03	4.0E-01	5.9E+01	**1.8E-03**
	90	4.9E+06	4.9E-01	8.9E+01	7.7E-02	**1.2E-02**	8.9E+01	1.9E-01
F6	10	3.5E-02	7.5E-04	3.1E-03	6.7E-05	**3.5E-05**	3.2E-02	6.8E-04
	30	1.0E+02	2.3E-03	7.1E-02	1.2E-03	**1.8E-07**	2.8E+00	9.5E-04
	60	2.6E+03	1.8E-03	6.0E+00	**2.4E-04**	7.9E-03	1.0E+01	3.6E-02
	90	1.2E+04	1.5E-02	1.0E+01	**2.6E-03**	3.0E-02	1.7E+01	5.1E-02
F7	10	1.5E-02	1.2E-03	4.8E-04	**4.3E-04**	5.6E-04	4.9E-04	7.1E-04
	30	6.4E+00	**3.3E-04**	6.5E-04	4.4E-04	8.7E-04	4.4E-03	1.0E-03
	60	7.5E+01	4.0E-04	2.0E-03	3.5E-04	2.3E-03	1.8E-04	**1.5E-04**
	90	7.9E+02	3.5E-04	1.2E-02	**1.9E-04**	2.6E-04	2.1E-04	1.7E-03
F8	10	-2.5E+03	-4.2E+03	-3.1E+03	-4.2E+03	-4.2E+03	-3.3E+03	-4.2E+03
	30	-7.9E+03	-1.3E+04	-8.0E+03	-1.3E+04	-1.3E+04	**-7.9E+03**	-1.3E+04
	60	**-1.2E+04**	-2.5E+04	-1.4E+04	-2.5E+04	-2.5E+04	-1.2E+04	-2.5E+04
	90	-1.6E+04	-3.8E+04	-1.7E+04	-3.8E+04	-3.8E+04	**-1.5E+04**	-3.8E+04
F9	10	3.0E+01	1.1E-09	**0.0E+00**	2.6E-05	1.2E-05	9.3E-05	1.9E-07
	30	1.9E+02	2.0E-06	**0.0E+00**	7.7E-05	5.4E-04	8.1E-05	3.7E-05
	60	3.6E+02	9.1E-06	**0.0E+00**	5.8E-04	8.5E-05	1.2E-04	1.7E-03
	90	6.7E+02	8.2E-05	**0.0E+00**	1.3E-04	3.6E-04	5.8E-04	3.2E-05
F10	10	4.2E-01	9.8E-04	**8.9E-16**	5.7E-04	2.9E-03	1.7E-04	3.7E-03
	30	1.9E+01	7.8E-04	**8.9E-16**	6.8E-03	1.9E-02	1.9E-03	1.3E-02
	60	1.9E+01	8.2E-04	**8.9E-16**	8.7E-04	1.3E-03	3.7E-04	1.2E-03
	90	1.9E+01	1.0E-03	**8.9E-16**	4.5E-03	1.7E-02	1.5E-03	2.6E-03
F11	10	2.8E-01	4.5E-05	**0.0E+00**	6.2E-03	7.8E-03	3.6E-04	8.4E-06
	30	1.9E+00	5.3E-05	**0.0E+00**	7.8E-03	2.7E-03	4.5E-03	4.7E-03
	60	2.2E+01	3.5E-05	**0.0E+00**	3.5E-02	1.6E-02	1.5E-06	3.2E-03
	90	1.0E+02	**0.0E+00**	**0.0E+00**	6.8E-03	8.3E-02	4.2E-04	3.8E-04
F12	10	1.9E-02	7.1E-04	8.2E-05	**1.5E-05**	5.7E-04	8.7E-03	1.2E-04
	30	1.4E+01	9.1E-05	7.6E-03	2.9E-06	**8.6E-07**	9.5E-02	9.7E-07
	60	1.0E+01	3.3E-05	7.7E-02	1.2E-05	6.8E-04	3.1E-01	**4.1E-06**
	90	3.7E+01	1.3E-05	2.0E-01	**2.7E-06**	6.6E-05	4.1E-01	2.4E-05
	10	2.1E-02	1.9E-03	1.3E-04	1.4E-04	**2.3E-05**	1.4E-02	8.1E-04

TABLE 8. ***(Continued.)*** **The best results obtained by the E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, E6GOA and GOA on the test optimization functions in (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.**

F13	30	3.1E+01	2.1E-04	3.0E+00	2.3E-03	3.6E-04	1.3E+00	**4.9E-05**
	60	9.7E+02	**9.7E-06**	6.0E+00	1.6E-03	3.3E-04	5.5E+00	9.7E-06
	90	2.9E+06	1.0E-04	9.0E+00	3.2E-04	**3.0E-05**	8.4E+00	4.1E-04
F14	2	2.0E+00	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**
F15	4	1.0E+00	1.4E+00	9.7E-01	**9.7E-01**	1.0E+00	9.7E-01	9.7E-01
F16	2	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**
F17	2	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**
F18	2	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**
F19	3	-6.8E-02	**-1.9E+00**	-6.8E-02	**-1.9E+00**	**-1.9E+00**	-6.8E-02	**-1.9E+00**
F20	6	-5.1E-03	**-1.2E+00**	-5.1E-03	**-1.2E+00**	**-1.2E+00**	-5.1E-03	**-1.2E+00**
F21	4	-5.1E+00	**-1.0E+01**	-5.1E+00	**-1.0E+01**	**-1.0E+01**	-5.1E+00	**-1.0E+01**
F22	4	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**
F23	4	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**
F24	10	6.5E-01	1.0E-08	**0.0E+00**	5.0E-06	2.4E-04	2.3E-04	4.6E-06
	30	5.7E+02	5.6E-06	**0.0E+00**	4.4E-03	7.8E-04	1.3E-03	1.4E-02
	60	3.2E+04	**4.2E-06**	5.3E-04	3.8E-03	7.4E-02	1.7E-02	1.5E-02
	90	3.1E+05	5.1E-03	**4.2E-05**	3.2E-01	3.3E-03	4.5E-01	1.4E-03
F25	10	8.2E-01	1.8E-01	6.7E-01	1.9E-01	1.9E-01	6.7E-01	**1.1E-01**
	30	3.9E+01	2.5E-01	7.7E-01	2.5E-01	**2.5E-01**	9.6E-01	2.5E-01
	60	1.2E+04	**2.5E-01**	9.2E-01	2.6E-01	2.6E-01	9.9E-01	2.6E-01
	90	2.3E+05	**2.5E-01**	9.7E-01	2.5E-01	2.6E-01	1.0E+00	2.5E-01
F26	10	1.1E+01	7.2E-06	1.9E-04	5.2E-06	**2.7E-08**	7.7E-03	4.1E-06
	30	6.3E+01	3.3E-06	5.9E-01	**1.2E-07**	1.0E-06	6.6E-01	1.7E-06
	60	1.5E+02	9.8E-06	1.9E+00	5.7E-06	**8.9E-08**	2.9E+00	5.1E-06
	90	2.4E+02	1.8E-04	3.7E+00	**1.8E-08**	6.6E-06	5.6E+00	2.6E-05
F27	10	**-7.1E+01**	**-7.1E+01**	**-7.1E+01**	**-7.1E+01**	**-7.1E+01**	**-7.1E+01**	**-7.1E+01**
	30	-1.7E+01	-2.5E+02	-2.6E+02	-2.5E+02	**-2.5E+02**	-1.2E+02	-2.5E+02
	60	3.1E+01	-5.3E+02	-5.3E+02	-5.3E+02	-5.3E+02	**-6.6E+01**	-5.3E+02
	90	2.2E+02	-8.1E+02	-8.1E+02	-8.1E+02	-8.1E+02	**-9.1E+01**	-8.1E+02
F28	10	7.9E+00	9.4E-07	**0.0E+00**	9.0E-07	3.9E-07	1.2E-06	1.0E-06
	30	4.5E+01	2.8E-06	**0.0E+00**	3.6E-05	2.2E-04	2.7E-05	3.1E-06
	60	1.5E+02	4.8E-07	**0.0E+00**	6.9E-06	4.1E-05	6.0E-07	3.1E-06
	90	8.5E+02	8.2E-06	**0.0E+00**	1.2E-05	7.8E-04	2.6E-04	7.8E-04
F29	10	-4.4E+00	**-7.6E+00**	-4.5E+00	-7.1E+00	-5.9E+00	-6.2E+00	-6.3E+00
	30	**-1.1E+01**	-1.5E+01	-1.3E+01	-1.8E+01	-1.7E+01	-1.3E+01	-1.6E+01
	60	**-1.9E+01**	-2.7E+01	-2.4E+01	-2.6E+01	-2.4E+01	-2.0E+01	-2.9E+01
	90	**-2.2E+01**	-4.0E+01	-3.3E+01	-3.7E+01	-3.8E+01	-2.7E+01	-3.7E+01
F30	10	1.0E+00	3.0E-09	**0.0E+00**	3.4E-05	2.0E-05	1.9E-04	1.4E-05
	30	3.6E+02	3.4E-10	**0.0E+00**	8.5E-04	1.5E-02	3.1E-02	2.6E-05
	60	9.6E+02	6.2E-04	**0.0E+00**	8.0E-01	4.8E-03	4.7E-02	6.1E-01
	90	1.9E+03	8.9E-04	**0.0E+00**	3.2E+00	9.0E+00	2.0E-02	2.6E+00
F31	10	2.4E+04	1.8E+01	**0.0E+00**	2.6E+03	7.2E+03	7.7E+01	1.2E-01
	30	1.5E+08	5.4E+01	**0.0E+00**	1.8E-03	1.4E+03	5.5E+01	7.9E+02
	60	2.5E+09	1.6E+01	**1.5E+01**	9.1E+02	7.7E+02	6.6E+03	1.4E+04
	90	9.6E+09	1.4E+00	**0.0E+00**	6.0E+02	1.4E+04	3.8E+01	2.0E+05
F32	10	-3.5E+02	-3.9E+02	-3.5E+02	-3.9E+02	-3.9E+02	**-3.5E+02**	-3.9E+02
	30	**-9.6E+02**	-1.2E+03	-9.8E+02	-1.2E+03	-1.2E+03	-9.7E+02	-1.2E+03

TABLE 8. *(Continued.)* The best results obtained by the E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, E6GOA and GOA on the test optimization functions in (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.

	60	-1.7E+03	-2.4E+03	-2.0E+03	-2.4E+03	-2.4E+03	**-1.7E+03**	-2.4E+03
	90	**-2.1E+03**	-3.5E+03	-2.6E+03	-3.5E+03	-3.5E+03	-2.1E+03	-3.5E+03
F33	10	4.0E-04	4.1E-16	**0.0E+00**	1.4E-10	6.5E-11	1.9E-12	3.2E-13
	30	1.3E-01	5.8E-14	**0.0E+00**	4.1E-13	1.5E-11	2.0E-11	3.3E-12
	60	8.2E-01	3.6E-12	**0.0E+00**	2.0E-11	2.2E-12	9.0E-13	2.0E-12
	90	1.5E+00	1.6E-13	2.0E-14	3.5E-14	3.3E-11	**7.6E-16**	3.6E-13
F34	10	1.3E-01	2.5E-05	**0.0E+00**	2.9E-03	4.8E-05	4.9E-03	5.5E-05
	30	3.6E+02	6.2E-05	**3.5E-05**	3.4E-03	2.3E-02	7.3E-03	5.3E-03
	60	2.2E+04	6.0E-04	**0.0E+00**	2.6E-01	6.2E-02	1.3E-01	3.2E-02
	90	1.3E+05	2.3E-04	**0.0E+00**	1.9E-02	8.6E-01	2.3E-03	5.4E-01

obviously better convergence and final optimum value in these functions as well. Figures 6(b, c) show an extremely better performance for E2GOA in the first iterations. The rate of convergence for GOA is similar in Figures 6(b, c), as shown GOA seems to trap quickly into a local optimum. Figure 6(d) show similar results to the above cases for the functions Levy function with 90 dimensions. Figure 6 as a whole show also that all the proposed algorithms are stable in its outstanding performance over several dimensionalities of the benchmark functions. We can conclude that all the proposed algorithms are indeed superior to the original GOA algorithm. However, that the performance of the six proposed algorithms is different from one function to another. An algorithm may outperform the other, depending on the type of application that uses the algorithm. This is what prompted us to present them where their performance could be different in a specific application.

B. SCENARIO 2: PERFORMANCE OF EGOA AGAINST OLD OPTIMIZATION ALGORITHMS

The experimental test of scenario 2 verified the effectiveness of the six proposals for an enhanced GOA against old optimization algorithms, which are ABC, BAT, CS, HS, and PSO. These comparisons are benchmarked using the same group of 34 test optimization functions listed in Table 4. The benchmarks are divided into 2 types that can evaluate the different capabilities of the algorithms: Benchmarks F1-F13 and F24-F34 of the low-dimensional optimization functions that are set to 2, 3,4, and 6, Benchmarks F14-F23 of the high-dimensional optimization functions that are set to 10, 30, 60, and 90 listed in Table 10. The dimensions of the benchmark functions are modified in order to test the scalability of the proposed algorithms against higher dimensional optimization functions. The algorithms were run 30 times using 50 generations each run, with random seeds. In this same manner, the set of test optimization functions are also kept the same as previously stated. However, the compared algorithms are six metaheuristic algorithms selected from two fields: evolutionary algorithms and swarm intelligence. To put the performance of the six proposals for enhancing GOA (EGOA) algorithms in perspective and illustrate its merits among similar metaheuristic methods, we compare its performance on global numeric optimization problems with six other algorithms. All the important parameters for the involved algorithms are listed in Table 9.

TABLE 9. Parameters of the six old algorithms used for performance analysis the proposed algorithms.

Alg.	Parameter	Symbol / Abbr.	Value
ABC	Food Number	*Fn*	100
BAT	Frequency minimum	*Qmin*	0
	Frequency maximum	*Qmax*	2
	Alfa	-	[0, 1]
	Gama	-	[0, 1]
PSO	Inertial constant	-	0.3
	Cognitive constant	-	1
	Social constant for swarm interaction	-	1
HS	Harmony Memory Size	*HMS*	50
	Harmony Memory Consideration Rate	*HMCR*	0.95
	Pitch Adjustment Rate	*PAR*	0.1
CS	Rate of alien eggs/solutions	*pa*	0.25

Table 10 presents the best possible optimum solution obtained by various EGOAs and the new metaheuristic algorithms on 2, 3, 4, and 6 of the low dimensional optimization functions. It can be seen from the table that the E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA performance is slightly less superior to other algorithms, except the E2GOA algorithm that shows better statistical results compared to the other algorithms on the low dimensional optimization functions. The E2GOA algorithm also shows the best performance, especially in terms of the quality of the reached optimum value in most of the cases.

Moreover, Table 10 presents the results on F1- F13 and F24 - F34 in 10, 30, 60, and 90 of the high-dimensional optimization functions. It was the fastest among the algorithms to get the global optimum on functions: F1, F3, F4, F5, F6, F7, F9, F11, F12, F13, F24, F25, F28, F30, and F33, with the exception of some of the functions, that the proposed algorithm has not been able to win in all statistical measurements (best, mean, and standard deviation), particularly functions F2, F8, F10, F26, F27, F29, F31, and F32.

TABLE 10. The best, mean, and standard deviation of test function values found by the ABC, BAT, CS, PSO, HS, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, AND E6GOA on the test optimization functions with (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.

F	D	Statistics	ABC	BAT	CS	PSO	HS	E1GOA	E2GOA	E3GOA	E4GOA	E5GOA	E6GOA
F1	10	Best	3.2E-02	8.0E+03	6.1E+02	9.7E-01	3.3E+00	3.0E-04	**0.0E+00**	1.1E-02	2.9E-04	1.3E-04	5.2E-03
		Mean	7.8E-01	1.1E+04	1.0E+03	3.5E+02	3.2E+01	5.9E-02	**4.3E-16**	9.1E-02	6.5E-02	5.4E-02	4.9E-02
		Std. dev	1.3E+00	9.9E+02	3.8E+02	7.5E+02	5.2E+01	6.2E-02	**3.4E-16**	7.6E-02	1.1E-01	6.2E-02	3.4E-02
	30	Best	6.5E+03	4.7E+04	1.3E+04	5.0E+01	5.9E+01	1.4E-08	**0.0E+00**	1.9E-05	1.6E-01	2.3E-06	2.2E-02
		Mean	9.6E+03	4.7E+04	1.7E+04	3.1E+03	8.6E+02	6.4E-01	**9.2E-16**	4.0E-01	8.9E-01	4.1E-01	7.2E-01
		Std. dev	2.5E+03	2.0E+01	3.7E+03	4.8E+03	1.6E+03	1.2E+00	**6.7E-16**	5.7E-01	7.3E-01	3.9E-01	7.0E-01
	60	Best	6.0E+04	1.0E+05	3.2E+04	3.8E+02	2.7E+03	1.0E-03	**5.8E-06**	5.1E-04	5.3E-02	3.3E-02	8.4E-05
		Mean	7.0E+04	1.4E+05	4.9E+04	1.2E+04	7.6E+03	**3.2E-01**	1.1E+00	4.6E-01	1.4E+00	7.8E-01	5.5E-01
		Std. dev	7.4E+03	1.5E+04	1.0E+04	1.8E+04	6.9E+03	**4.4E-01**	1.5E+00	4.7E-01	1.5E+00	1.1E+00	9.1E-01
	90	Best	1.1E+05	1.5E+05	6.4E+04	1.2E+03	1.0E+04	1.8E-05	**0.0E+00**	3.8E-05	2.8E-04	2.1E-06	3.8E-02
		Mean	1.4E+05	1.5E+05	8.3E+04	2.4E+04	2.2E+04	**4.4E-01**	3.9E+00	1.8E+00	1.1E+00	5.3E-01	1.3E+00
		Std. dev	1.5E+04	5.4E+00	1.1E+04	3.1E+04	1.9E+04	6.3E-01	5.0E+00	2.1E+00	1.4E+00	5.7E-01	1.2E+00
F2	10	Best	5.0E-02	2.4E+01	8.6E+00	3.1E-01	3.9E-02	2.4E-04	**0.0E+00**	3.4E-03	6.4E-03	2.3E-03	8.7E-03
		Mean	2.1E-01	2.4E+01	1.2E+01	4.0E+00	8.8E-02	5.8E-02	**9.7E-03**	3.2E-02	5.7E-02	5.2E-02	1.2E-01
		Std. dev	9.8E-02	**2.4E-15**	2.0E+00	4.3E+00	1.1E-01	7.2E-02	1.6E-02	4.5E-02	5.7E-02	9.7E-02	1.2E-01
	30	Best	3.1E+01	1.2E+02	5.7E+01	6.4E+00	2.0E+00	2.1E-03	**0.0E+00**	1.4E-02	2.2E-02	2.9E-03	1.5E-03
		Mean	3.8E+01	1.2E+02	7.8E+01	4.0E+01	6.2E+00	1.9E-01	**1.3E-08**	3.0E-01	2.4E-01	1.1E-01	1.5E-01
		Std. dev	3.8E+00	**0.0E+00**	9.9E+00	4.2E+01	6.7E+00	2.6E-01	5.7E-09	3.7E-01	2.3E-01	1.2E-01	2.2E-01
	60	Best	1.2E+02	2.4E+02	1.6E+02	2.4E+02	1.2E+01	2.0E-03	**0.0E+00**	2.8E-02	8.4E-02	3.2E-02	6.9E-02
		Mean	1.5E+02	2.4E+02	1.8E+02	2.4E+02	2.8E+01	4.0E-01	**1.8E-08**	4.6E-01	6.0E-01	4.4E-01	6.3E-01
		Std. dev	1.6E+01	1.3E-11	1.3E+01	**0.0E+00**	1.9E+01	6.0E-01	1.1E-08	4.6E-01	6.0E-01	4.0E-01	6.3E-01
	90	Best	2.8E+02	3.7E+02	2.4E+02	3.7E+02	3.9E+01	2.1E-03	**0.0E+00**	7.1E-02	1.3E-02	1.1E-02	1.1E-02
		Mean	3.0E+02	3.7E+02	2.9E+02	3.7E+02	7.1E+01	4.5E-01	**2.5E-08**	4.8E-01	9.3E-01	3.5E-01	8.6E-01
		Std. dev	1.1E+01	9.5E-14	2.2E+01	**0.0E+00**	4.0E+01	7.5E-01	1.2E-08	4.6E-01	1.1E+00	4.3E-01	1.3E+00
F3	10	Best	8.6E+02	6.7E+03	9.4E+02	4.0E+01	4.3E+02	2.2E-07	**0.0E+00**	3.1E-02	9.7E-02	1.8E-02	2.6E-03
		Mean	2.5E+03	6.7E+03	2.2E+03	2.6E+03	8.6E+02	1.2E-01	**2.1E-03**	5.5E-01	1.2E+00	7.2E-01	2.5E+00
		Std. dev	1.3E+03	**5.9E-06**	7.1E+02	3.5E+03	9.4E+02	1.6E-01	3.4E-03	6.4E-01	1.2E+00	8.9E-01	5.2E+00
	30	Best	3.0E+04	9.3E+04	2.6E+04	3.9E+04	1.2E+04	9.2E-05	**0.0E+00**	2.9E-01	8.2E-03	2.5E-01	5.6E-01
		Mean	4.0E+04	1.0E+05	4.3E+04	3.0E+05	2.6E+04	9.5E+00	**1.3E-15**	1.3E+01	1.7E+01	9.2E+00	2.5E+02
		Std. dev	7.4E+03	3.8E+03	6.7E+03	2.2E+05	9.6E+03	1.9E+01	**5.8E-16**	1.1E+01	3.2E+01	1.9E+01	2.7E+02
	60	Best	1.3E+05	3.6E+05	1.4E+05	1.1E+06	1.8E+05	8.6E-05	**0.0E+00**	2.9E+00	2.4E-02	2.6E-02	1.9E-01
		Mean	1.6E+05	3.6E+05	1.6E+05	5.4E+06	2.0E+05	6.6E+00	**5.9E-15**	2.1E+02	9.5E+01	7.5E+00	8.6E+02
		Std. dev	1.7E+04	1.4E+00	1.9E+04	3.8E+06	1.6E+04	1.9E+01	**4.3E-15**	2.5E+02	1.6E+02	1.7E+01	1.5E+03
	90	Best	2.3E+05	5.7E+05	2.4E+05	2.5E+06	3.5E+05	1.0E-04	**0.0E+00**	4.9E-01	1.0E+01	3.2E+00	2.2E+00
		Mean	3.2E+05	5.7E+05	3.1E+05	2.4E+07	4.0E+05	3.5E+01	**9.0E-15**	3.4E+02	8.1E+01	2.0E+01	2.6E+04
		Std. dev	6.2E+04	2.9E+01	3.9E+04	1.4E+07	8.5E+04	4.8E+01	**8.8E-15**	4.3E+02	8.2E+01	1.4E+01	3.3E+04
F4	10	Best	2.1E+01	6.1E+01	2.4E+01	1.2E+00	1.3E+00	1.2E-02	**0.0E+00**	1.3E-02	4.9E-03	7.7E-03	6.0E-03
		Mean	2.7E+01	7.1E+01	3.1E+01	8.6E+00	9.5E+00	1.3E-01	**2.1E-02**	1.3E-01	7.6E-02	1.2E-01	1.0E-01
		Std. dev	4.5E+00	3.6E+00	4.6E+00	8.7E+00	9.0E+00	9.6E-02	6.7E-02	1.4E-01	**6.1E-02**	1.2E-01	8.4E-02
	30	Best	6.6E+01	6.2E+01	4.4E+01	2.2E+01	2.0E+01	2.1E-03	**0.0E+00**	8.5E-03	1.1E-02	3.1E-03	1.6E-03
		Mean	7.2E+01	6.3E+01	5.6E+01	4.1E+01	3.0E+01	1.5E-01	**7.0E-05**	1.4E-01	1.2E-01	1.8E-01	7.2E-02
		Std. dev	4.4E+00	2.6E-01	6.1E+00	1.9E+01	1.2E+01	1.5E-01	**2.2E-04**	1.3E-01	9.1E-02	1.8E-01	7.9E-02
	60	Best	8.6E+01	8.5E+01	5.6E+01	9.2E+01	3.9E+01	7.1E-04	**0.0E+00**	3.7E-02	2.4E-04	1.0E-02	1.6E-02
		Mean	8.9E+01	9.0E+01	6.9E+01	9.2E+01	5.3E+01	**6.2E-02**	1.6E-01	1.9E-01	1.1E-01	1.7E-01	2.4E-01
		Std. dev	1.5E+00	1.8E+00	7.6E+00	**0.0E+00**	1.3E+01	9.1E-02	4.4E-01	1.1E-01	9.9E-02	1.5E-01	2.2E-01
	90	Best	9.3E+01	9.1E+01	7.2E+01	9.6E+01	5.6E+01	1.6E-03	**0.0E+00**	8.8E-03	1.1E-03	2.3E-02	4.3E-02
		Mean	9.5E+01	9.2E+01	8.0E+01	9.6E+01	6.9E+01	**1.1E-01**	2.1E-01	2.2E-01	1.4E-01	1.6E-01	1.7E-01
		Std. dev	8.4E-01	3.3E-02	4.8E+00	0.0E+00	9.7E+00	1.1E-01	2.3E-01	1.7E-01	8.3E-02	8.6E-02	1.6E-01
F5	10	Best	3.1E+01	1.6E+07	8.1E+04	6.0E+01	2.4E+01	3.9E-02	8.3E+00	7.9E-03	1.9E-03	8.9E+00	**1.4E-03**
		Mean	1.5E+02	2.2E+07	2.3E+05	9.3E+04	4.3E+03	**3.9E+00**	9.0E+00	4.2E+00	4.3E+00	9.5E+00	4.8E+00
		Std. dev	1.4E+02	2.1E+06	9.1E+04	2.7E+05	1.2E+04	4.6E+00	**6.7E-01**	4.4E+00	5.0E+00	7.9E-01	4.6E+00
	30	Best	9.6E+05	8.3E+07	9.9E+06	1.7E+02	4.2E+03	9.0E-02	2.9E+01	**2.6E-05**	2.3E-03	2.9E+01	5.5E-04
		Mean	1.1E+07	8.5E+07	2.5E+07	1.8E+06	8.0E+05	1.9E+01	3.0E+01	1.3E+01	**5.8E+00**	3.3E+01	1.8E+01
		Std. dev	7.4E+06	5.1E+05	9.4E+06	4.2E+06	2.0E+06	1.6E+01	**3.3E+00**	1.7E+01	9.3E+00	6.6E+00	1.9E+01
	60	Best	1.6E+08	3.3E+08	5.0E+07	1.7E+04	8.3E+05	3.4E-02	5.9E+01	6.8E-03	4.0E-01	5.9E+01	**1.8E-03**
		Mean	2.7E+08	3.3E+08	1.0E+08	1.3E+07	1.1E+07	3.4E+01	5.9E+01	2.6E+01	3.1E+01	6.9E+01	**2.3E+01**
		Std. dev	4.8E+07	9.6E+03	3.5E+07	2.9E+07	1.8E+07	3.4E+01	**1.1E-01**	3.1E+01	3.2E+01	1.2E+01	3.2E+01
	90	Best	4.2E+08	2.8E+08	1.2E+08	9.6E+04	8.9E+06	4.9E-01	8.9E+01	7.7E-02	**1.2E-02**	8.9E+01	1.9E-01
		Mean	5.9E+08	2.9E+08	1.8E+08	7.1E+07	3.8E+07	3.0E+01	1.0E+02	6.2E+01	**3.0E+01**	9.7E+01	5.0E+01
		Std. dev	8.2E+07	5.8E+06	4.3E+07	1.4E+08	4.5E+07	4.5E+01	3.0E+01	5.1E+01	4.1E+01	**7.0E+00**	4.8E+01
F6	10	Best	3.3E+00	1.0E+04	4.2E+02	4.2E-01	4.5E+00	7.5E-04	3.1E-03	6.7E-05	**3.5E-05**	3.2E-02	6.8E-04
		Mean	4.3E+00	1.0E+04	1.1E+03	4.3E+02	2.2E+01	1.0E-01	**3.2E-02**	7.6E-02	8.1E-02	1.4E-01	1.2E-01
		Std. dev	9.2E-01	4.4E+01	5.1E+02	9.9E+02	2.1E+01	8.1E-02	**3.6E-02**	8.0E-02	8.5E-02	7.2E-02	1.3E-01
	30	Best	8.1E+03	3.7E+04	1.2E+04	6.1E+01	5.1E+01	2.3E-03	7.1E-02	1.2E-03	1.8E-07	2.8E+00	9.5E-04
		Mean	1.2E+04	4.4E+04	1.8E+04	3.4E+03	1.6E+03	5.1E-01	7.2E-01	**2.4E-01**	2.1E+00	4.7E+00	5.2E-01

TABLE 10. ***(Continued.)*** **The best, mean, and standard deviation of test function values found by the ABC, BAT, CS, PSO, HS, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, AND E6GOA on the test optimization functions with (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.**

		Std. dev	2.5E+03	2.6E+03	4.5E+03	6.3E+03	2.9E+03	7.3E-01	9.8E-01	**3.5E-01**	3.5E+00	1.3E+00	5.4E-01
	60	Best	6.7E+04	1.1E+05	4.1E+04	8.9E+02	2.5E+03	1.8E-03	6.0E+00	2.4E-04	7.9E-03	1.1E+01	3.6E-02
		Mean	8.2E+04	1.4E+05	5.2E+04	1.2E+04	7.6E+03	**5.8E-01**	1.2E+01	1.2E+00	6.9E-01	1.2E+01	9.2E-01
		Std. dev	6.3E+03	8.3E+03	6.4E+03	1.8E+04	6.9E+03	**4.9E-01**	6.0E+00	1.6E+00	5.0E-01	1.2E+00	1.1E+00
	90	Best	1.4E+05	1.5E+05	7.0E+04	2.2E+03	1.1E+04	1.5E-02	1.0E+01	2.6E-03	3.1E-02	1.7E+01	5.1E-02
		Mean	1.6E+05	1.5E+05	8.0E+04	4.8E+04	2.2E+04	1.2E+00	1.5E+01	1.3E+00	**7.4E-01**	1.9E+01	1.2E+00
		Std. dev	9.5E+03	7.8E-01	5.2E+03	7.2E+04	1.9E+04	1.9E+00	4.3E+00	1.1E+00	**6.3E-01**	1.3E+00	8.3E-01
F7	10	Best	2.1E+00	4.4E-01	9.6E-02	6.0E-02	3.3E-04	1.2E-03	4.8E-04	4.3E-04	5.6E-04	4.9E-04	7.1E-04
		Mean	2.8E+00	5.0E-01	2.4E-01	2.9E-01	**3.7E-03**	6.7E-03	1.5E-02	1.3E-02	7.0E-03	1.7E-02	1.7E-02
		Std. dev	4.0E-01	8.6E-02	9.2E-02	3.7E-01	**3.3E-03**	6.6E-03	9.4E-03	1.2E-02	7.2E-03	1.2E-02	1.5E-02
	30	Best	2.1E+01	1.1E+02	1.5E+01	7.0E-02	7.7E-03	**3.3E-04**	6.5E-04	4.4E-04	8.7E-04	4.4E-03	1.0E-03
		Mean	3.4E+01	1.1E+02	2.2E+01	1.5E+00	6.0E-02	**8.5E-03**	3.0E-02	1.5E-02	1.3E-02	2.5E-02	1.8E-02
		Std. dev	1.0E+01	2.1E-02	5.6E+00	2.4E+00	9.5E-02	**9.1E-03**	2.6E-02	1.3E-02	1.3E-02	2.1E-02	1.5E-02
	60	Best	4.6E+02	2.5E+02	9.6E+01	3.1E-02	2.3E-01	4.0E-04	2.0E-03	3.5E-04	2.3E-03	1.8E-04	**1.5E-04**
		Mean	5.4E+02	3.7E+02	1.9E+02	1.8E+01	6.0E+00	6.0E-03	5.9E-02	**5.0E-03**	1.3E-02	1.4E-02	1.4E-02
		Std. dev	5.3E+01	4.3E+01	4.7E+01	4.5E+01	1.4E+01	1.1E-02	1.0E-01	**8.2E-03**	1.9E-02	1.1E-02	1.8E-02
	90	Best	1.5E+03	5.9E+02	2.7E+02	8.1E-01	3.6E+00	3.5E-04	1.2E-02	**1.9E-04**	2.6E-04	2.1E-04	1.7E-03
		Mean	1.6E+03	8.6E+02	4.8E+02	5.4E+01	3.8E+01	1.3E-02	6.0E-02	**4.3E-03**	1.0E-02	2.1E-02	6.2E-03
		Std. dev	1.0E+02	9.3E+01	1.2E+02	1.2E+02	7.4E+01	2.1E-02	4.9E-02	**4.7E-03**	7.8E-03	2.0E-02	5.7E-03
F8	10	Best	8.6E+02	1.4E+03	2.6E+03	2.3E+03	3.8E+03	3.8E+03	2.7E+03	3.8E+03	3.8E+03	2.9E+03	3.8E+03
		Mean	**1.0E+03**	1.2E+03	2.3E+03	1.9E+03	3.8E+03	3.8E+03	2.7E+03	3.8E+03	3.8E+03	2.7E+03	3.8E+03
		Std. dev	5.2E+02	5.0E+02	5.3E+02	6.2E+02	4.5E+02	4.2E+02	**4.2E+02**	4.2E+02	4.2E+02	5.0E+02	4.2E+02
	30	Best	5.5E+03	3.2E+03	5.1E+03	6.6E+03	1.2E+04	1.2E+04	7.6E+03	1.2E+04	1.2E+04	7.5E+03	1.2E+04
		Mean	6.0E+03	**3.2E+03**	4.8E+03	4.8E+03	1.1E+04	1.2E+04	7.5E+03	1.2E+04	1.2E+04	7.2E+03	1.2E+04
		Std. dev	7.2E+02	4.3E+02	5.4E+02	1.7E+03	1.3E+03	4.2E+02	4.9E+02	4.2E+02	4.2E+02	7.4E+02	**4.2E+02**
	60	Best	1.5E+04	2.4E+03	8.4E+03	1.7E+04	2.3E+04	2.5E+04	1.4E+04	2.5E+04	2.5E+04	1.2E+04	2.5E+04
		Mean	1.6E+04	**2.4E+03**	7.4E+03	1.1E+04	2.2E+04	2.5E+04	1.3E+04	2.5E+04	2.5E+04	1.1E+04	2.5E+04
		Std. dev	8.9E+02	4.2E+02	8.5E+02	4.8E+03	2.4E+03	4.2E+02	1.1E+03	4.2E+02	4.2E+02	1.1E+03	**4.2E+02**
	90	Best	2.5E+04	4.6E+03	9.9E+03	1.4E+04	3.3E+04	3.7E+04	1.7E+04	3.7E+04	3.7E+04	1.5E+04	3.7E+04
		Mean	2.6E+04	**4.6E+03**	9.2E+03	9.6E+03	3.1E+04	3.7E+04	1.5E+04	3.7E+04	3.7E+04	1.4E+04	3.7E+04
		Std. dev	9.9E+02	**4.2E+02**	7.3E+02	3.5E+03	4.3E+03	4.2E+02	1.5E+03	4.2E+02	4.2E+02	1.5E+03	4.2E+02
F9	10	Best	2.6E+00	5.3E+01	4.3E+01	4.2E+01	**0.0E+00**	1.1E-09	**0.0E+00**	2.6E-05	1.2E-05	9.3E-05	1.9E-07
		Mean	7.1E+00	5.3E+01	4.5E+01	4.6E+01	1.3E-01	3.9E-04	**0.0E+00**	1.0E+00	3.6E-01	2.2E-03	1.2E+00
		Std. dev	2.8E+00	5.9E-05	2.0E+00	5.2E+00	4.1E-01	6.7E-04	**0.0E+00**	3.1E+00	1.1E+00	3.9E-03	3.1E+00
	30	Best	9.3E+01	1.9E+02	1.9E+02	1.9E+02	6.4E+00	2.0E-06	**0.0E+00**	7.7E-05	5.4E-04	8.1E-05	3.7E-05
		Mean	1.1E+02	1.9E+02	1.9E+02	1.9E+02	1.1E+01	3.0E+00	0.0E+00	9.7E-02	6.1E+00	1.0E-01	3.1E+00
		Std. dev	1.1E+01	2.1E-04	2.5E-03	0.0E+00	6.6E+00	9.4E+00	**0.0E+00**	1.5E-01	1.3E+01	1.4E-01	9.4E+00
	60	Best	2.9E+02	4.1E+02	4.1E+02	4.1E+02	4.6E+01	9.1E-06	**0.0E+00**	5.8E-04	8.5E-05	1.2E-04	1.7E-03
		Mean	3.2E+02	4.1E+02	4.1E+02	4.1E+02	7.4E+01	8.1E-02	**2.2E-03**	6.1E+00	6.0E+00	2.5E-01	1.5E+01
		Std. dev	1.9E+01	5.1E-03	**0.0E+00**	**0.0E+00**	4.4E+01	2.2E-01	2.4E-03	1.9E+01	1.9E+01	5.2E-01	2.5E+01
	90	Best	4.9E+02	6.3E+02	6.3E+02	6.3E+02	1.2E+02	8.2E-05	**0.0E+00**	1.3E-04	3.6E-04	5.8E-04	3.2E-05
		Mean	5.4E+02	6.3E+02	6.3E+02	6.3E+02	1.8E+02	4.6E+00	**0.0E+00**	9.1E+00	1.9E-01	8.8E-01	2.8E+01
		Std. dev	2.8E+01	1.8E-05	**0.0E+00**	**0.0E+00**	8.1E+01	1.2E+01	**0.0E+00**	2.8E+01	2.8E-01	2.3E+00	4.2E+01
F10	10	Best	4.2E+00	1.8E+01	1.3E+01	1.3E+00	7.6E-01	9.8E-04	**8.9E-16**	5.7E-04	2.9E-03	1.7E-04	3.7E-03
		Mean	5.7E+00	1.8E+01	1.6E+01	6.6E+00	2.3E+00	1.9E-01	**9.8E-09**	2.7E-01	2.8E-01	9.0E-02	1.5E-01
		Std. dev	9.9E-01	2.0E-05	1.7E+00	4.6E+00	1.6E+00	2.7E-01	**4.3E-09**	3.6E-01	4.3E-01	1.4E-01	1.5E-01
	30	Best	1.7E+01	1.9E+01	1.9E+01	4.8E+00	4.0E+00	7.8E-04	**8.9E-16**	6.8E-03	1.9E-02	1.9E-03	1.4E-02
		Mean	1.8E+01	1.9E+01	1.9E+01	1.1E+01	6.3E+00	**8.7E-02**	2.5E-01	3.5E-01	2.5E-01	1.5E-01	3.5E-01
		Std. dev	2.9E-01	**7.1E-07**	1.3E-01	5.3E+00	3.2E+00	9.5E-02	4.3E-01	2.9E-01	2.1E-01	1.8E-01	3.4E-01
	60	Best	1.9E+01	1.9E+01	1.9E+01	1.9E+01	8.4E+00	8.2E-04	**8.9E-16**	8.7E-04	1.3E-03	3.8E-04	1.2E-03
		Mean	1.9E+01	1.9E+01	1.9E+01	1.9E+01	1.2E+01	**1.3E-01**	1.3E-01	2.1E-01	1.8E-01	2.3E-01	3.1E-01
		Std. dev	5.0E-02	1.2E-06	**3.7E-15**	**3.7E-15**	3.0E+00	2.3E-01	1.4E-01	2.3E-01	2.1E-01	1.7E-01	1.9E-01
	90	Best	1.9E+01	1.9E+01	1.9E+01	1.9E+01	1.1E+01	1.1E-03	**8.9E-16**	4.5E-03	1.7E-02	1.5E-03	2.6E-03
		Mean	1.9E+01	1.9E+01	1.9E+01	1.9E+01	1.4E+01	7.5E-02	**2.2E-02**	2.5E-01	1.6E-01	7.2E-02	3.0E-01
		Std. dev	5.3E-02	1.9E-07	**0.0E+00**	**0.0E+00**	2.7E+00	1.2E-01	6.9E-02	1.9E-01	1.4E-01	5.7E-02	1.4E-01
F11	10	Best	6.3E-01	6.7E+01	4.8E+00	6.6E-01	4.4E-01	4.5E-05	**0.0E+00**	6.2E-03	7.8E-03	3.6E-04	8.4E-06
		Mean	8.6E-01	6.7E+01	1.1E+01	4.0E+00	8.5E-01	2.7E-01	**1.4E-01**	1.8E-01	2.7E-01	2.0E-01	1.9E-01
		Std. dev	1.3E-01	**1.6E-02**	3.5E+00	6.7E+00	4.3E-01	2.0E-01	2.3E-01	1.4E-01	2.0E-01	1.8E-01	1.6E-01
	30	Best	7.9E+01	3.0E+02	9.0E+01	1.5E+00	1.7E+00	5.3E-05	**0.0E+00**	7.8E-03	2.7E-03	4.5E-03	4.7E-03
		Mean	1.1E+02	3.0E+02	1.6E+02	3.6E+01	1.1E+01	1.9E-01	**3.8E-15**	5.1E-01	2.2E-01	2.9E-01	5.0E-01
		Std. dev	2.3E+01	3.0E-02	3.8E+01	6.4E+01	1.8E+01	3.2E-01	**3.2E-15**	3.6E-01	2.6E-01	3.9E-01	3.2E-01
	60	Best	5.9E+02	1.0E+03	3.1E+02	9.0E+00	2.1E+01	3.5E-05	**0.0E+00**	3.5E-02	1.6E-02	1.5E-06	3.2E-03
		Mean	7.6E+02	1.0E+03	4.1E+02	2.6E+02	6.8E+01	4.0E-01	**2.5E-01**	5.1E-01	4.4E-01	4.0E-01	4.2E-01
		Std. dev	7.9E+01	**1.8E-02**	7.7E+01	4.5E+02	7.0E+01	3.7E-01	4.3E-01	3.5E-01	2.5E-01	2.8E-01	2.3E-01
	90	Best	1.3E+03	1.6E+03	6.3E+02	1.7E+01	8.1E+01	**0.0E+00**	**0.0E+00**	6.8E-03	8.3E-02	4.2E-04	3.8E-04
		Mean	1.5E+03	1.8E+03	7.8E+02	1.9E+02	2.0E+02	1.9E-01	**1.0E-01**	5.1E-01	4.8E-01	2.9E-01	2.3E-01
		Std. dev	9.4E+01	5.2E+01	1.3E+02	2.5E+02	1.7E+02	2.0E-01	**1.7E-01**	3.8E-01	3.3E-01	3.4E-01	2.8E-01
F12	10	Best	1.9E-03	1.9E+07	2.1E+01	3.5E-02	1.2E-01	7.1E-04	8.2E-05	**1.5E-05**	5.7E-04	8.7E-03	1.2E-04

TABLE 10. *(Continued.)* **The best, mean, and standard deviation of test function values found by the ABC, BAT, CS, PSO, HS, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, AND E6GOA on the test optimization functions with (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.**

		Mean	1.6E-01	1.9E+07	2.7E+03	2.8E+02	1.6E+00	9.8E-03	**1.6E-03**	4.5E-03	1.1E-02	8.0E-02	5.9E-03
		Std. dev	2.0E-01	1.4E+05	5.2E+03	8.7E+02	3.6E+00	7.8E-03	**2.4E-03**	5.7E-03	2.0E-02	7.0E-02	6.9E-03
	30	Best	6.3E+00	2.0E+08	5.7E+06	1.8E+00	1.4E+00	9.1E-05	7.6E-03	2.9E-06	**8.6E-07**	9.5E-02	9.7E-07
		Mean	8.3E+06	2.0E+08	2.0E+07	2.2E+06	4.1E+05	**1.2E-03**	4.7E-02	2.7E-03	3.0E-03	1.7E-01	1.6E-03
		Std. dev	9.9E+06	3.8E+00	1.3E+07	6.8E+06	1.2E+06	**1.1E-03**	4.8E-02	3.0E-03	3.0E-03	5.1E-02	1.6E-03
	60	Best	2.0E+08	8.0E+07	4.1E+07	4.9E+00	4.9E+03	3.3E-05	7.7E-02	1.2E-05	6.8E-04	3.1E-01	**4.1E-06**
		Mean	4.3E+08	4.0E+08	1.1E+08	3.9E+07	5.9E+06	**9.5E-04**	1.2E-01	1.2E-03	2.4E-03	4.3E-01	1.4E-03
		Std. dev	1.3E+08	1.1E+08	4.3E+07	1.1E+08	1.3E+07	**1.1E-03**	6.2E-02	1.1E-03	1.7E-03	7.5E-02	1.5E-03
	90	Best	1.1E+09	8.8E+08	1.3E+08	4.5E+00	1.1E+06	1.3E-05	2.0E-01	**2.7E-06**	6.6E-05	4.1E-01	2.4E-05
		Mean	1.4E+09	8.8E+08	2.7E+08	3.3E+08	3.7E+07	1.8E-03	2.9E-01	**8.4E-04**	1.7E-03	5.7E-01	1.2E-03
		Std. dev	1.5E+08	5.2E-01	1.2E+08	8.6E+08	7.8E+07	2.0E-03	7.9E-02	**8.9E-04**	2.7E-03	8.5E-02	1.3E-03
F13	10	Best	5.4E-03	2.6E+07	6.1E+03	6.5E-02	1.3E-01	1.9E-03	1.3E-04	1.4E-04	**2.3E-05**	1.4E-02	8.1E-04
		Mean	3.2E-02	2.7E+07	1.7E+05	2.8E+05	1.4E+01	2.5E-02	**2.4E-03**	1.8E-02	2.1E-02	3.7E-02	1.7E-02
		Std. dev	2.0E-02	2.6E+05	1.5E+05	8.8E+05	3.9E+01	2.0E-02	**3.6E-03**	2.3E-02	1.8E-02	1.9E-02	2.2E-02
	30	Best	2.5E+04	3.1E+08	2.9E+07	3.9E+00	5.8E+00	2.1E-04	3.0E+00	2.3E-03	3.6E-04	1.3E+00	**4.9E-05**
		Mean	1.7E+07	3.2E+08	7.3E+07	9.8E+06	5.4E+05	3.9E-02	3.0E+00	3.7E-02	2.6E-02	2.2E+00	**1.6E-02**
		Std. dev	1.5E+07	1.7E+06	2.0E+07	2.9E+07	1.5E+06	3.4E-02	**8.6E-03**	3.5E-02	2.2E-02	7.1E-01	1.6E-02
	60	Best	3.6E+08	1.3E+09	6.3E+07	5.4E+01	1.5E+05	**9.7E-06**	6.0E+00	1.6E-03	3.3E-04	5.5E+00	9.7E-06
		Mean	9.6E+08	1.3E+09	3.1E+08	3.7E+08	3.3E+07	**2.6E-02**	6.7E+00	4.4E-02	7.2E-02	6.3E+00	8.2E-02
		Std. dev	2.9E+08	5.4E+05	1.3E+08	1.0E+09	7.6E+07	**3.2E-02**	1.1E+00	3.6E-02	7.0E-02	6.1E-01	8.7E-02
	90	Best	2.3E+09	1.2E+09	3.8E+08	3.2E+03	1.5E+07	1.0E-04	9.0E+00	3.2E-04	**3.0E-05**	8.4E+00	4.1E-04
		Mean	2.6E+09	1.4E+09	5.8E+08	6.7E+08	1.3E+08	3.8E-02	9.0E+00	7.4E-02	**2.7E-02**	1.0E+01	4.0E-02
		Std. dev	2.8E+08	7.7E+07	1.7E+08	1.6E+09	1.9E+08	3.5E-02	**1.1E-02**	7.1E-02	3.5E-02	1.4E+00	4.3E-02
F14	2	Best	**1.0E+00**	8.2E+00	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**
		Mean	**1.0E+00**	1.1E+01	1.2E+00	1.0E+00	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**
		Std. dev	**9.8E-12**	8.0E-01	3.6E-01	2.1E-02	4.4E-10	2.2E-10	**9.8E-12**	4.2E-01	8.4E-01	1.4E+00	6.3E-01
F15	4	Best	1.6E+00	3.8E+00	1.4E+00	3.7E+00	4.2E+00	1.4E+00	9.7E-01	**9.7E-01**	1.0E+00	9.7E-01	9.7E-01
		Mean	2.5E+00	3.9E+00	2.5E+00	5.7E+00	4.2E+00	1.5E+00	1.4E+00	1.5E+00	1.3E+00	**1.3E+00**	1.4E+00
		Std. dev	7.4E-01	3.2E-02	7.5E-01	7.8E-01	**0.0E+00**	8.4E-02	2.7E-01	2.3E-01	1.9E-01	1.7E-01	2.0E-01
F16	2	Best	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**
		Mean	**-1.0E+00**	-9.7E-01	-1.0E+00	-1.0E+00	-1.0E+00	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**
		Std. dev	1.0E-08	2.1E-02	8.6E-06	9.9E-05	1.0E-03	2.0E-09	2.9E-09	5.9E-09	1.1E-09	3.0E-09	3.2E-09
F17	2	Best	**4.0E-01**	4.0E-01	**4.0E-01**	4.3E-01	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**
		Mean	4.0E-01	4.0E-01	4.0E-01	7.1E-01	4.0E-01	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**
		Std. dev	3.6E-07	3.2E-08	1.7E-04	2.3E-01	1.1E-04	1.7E-08	2.4E-09	1.1E-08	7.6E-09	2.8E-09	**2.3E-09**
F18	2	Best	**3.0E+00**	**3.0E+00**	**3.0E+00**	7.8E+00	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**
		Mean	**3.0E+00**	**3.0E+00**	**3.0E+00**	2.6E+01	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**
		Std. dev	**0.0E+00**	2.2E-14	**0.0E+00**	8.0E+00	3.7E-14	5.6E-15	**0.0E+00**	9.9E-15	8.4E-15	1.0E-14	1.3E-14
F19	3	Best	-6.8E-02	-1.9E+00	-6.8E-02	**-3.9E+00**	**-3.9E+00**	-1.9E+00	-6.8E-02	-1.9E+00	-1.9E+00	-6.8E-02	-1.9E+00
		Mean	-6.8E-02	-1.9E+00	-6.8E-02	-3.9E+00	-3.9E+00	-1.9E+00	-6.8E-02	-1.9E+00	-1.9E+00	-6.8E-02	-1.9E+00
		Std. dev	**0.0E+00**	3.0E-06	**0.0E+00**	5.6E-03	3.7E-03	1.9E-12	1.5E-17	1.9E-12	2.8E-12	1.5E-17	3.1E-11
F20	6	Best	-1.3E+00	-1.2E+00	-5.1E-03	-3.2E+00	**-3.3E+00**	-1.2E+00	-5.1E-03	-1.2E+00	-1.2E+00	-5.1E-03	-1.2E+00
		Mean	-1.3E+00	-1.2E+00	-5.1E-03	-3.0E+00	-3.3E+00	-1.2E+00	-5.1E-03	-1.2E+00	-1.2E+00	-5.1E-03	-1.2E+00
		Std. dev	**0.0E+00**	1.5E-04	9.1E-19	9.1E-02	6.0E-02	**0.0E+00**	9.1E-19	2.0E-12	2.9E-12	9.1E-19	6.4E-12
F21	4	Best	**-1.0E+01**	-9.6E-01	-8.6E+00	**-1.0E+01**	-2.7E+00	**-1.0E+01**	-5.1E+00	**-1.0E+01**	**-1.0E+01**	-5.1E+00	**-1.0E+01**
		Mean	-1.0E+01	-5.9E-01	-6.7E+00	-8.7E+00	-2.7E+00	**-1.0E+01**	-5.1E+00	**-1.0E+01**	**-1.0E+01**	-5.1E+00	**-1.0E+01**
		Std. dev	2.4E-01	1.3E-01	1.4E+00	2.1E+00	4.5E-02	6.4E-06	**4.6E-07**	1.8E-05	7.4E-06	4.5E-06	5.6E-06
F22	4	Best	**-1.0E+01**	-2.1E+00	-9.2E+00	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**
		Mean	-1.0E+01	-1.3E+00	-7.1E+00	-7.3E+00	-1.0E+01	-1.0E+01	-9.1E+00	-9.9E+00	**-1.0E+01**	-8.4E+00	**-1.0E+01**
		Std. dev	5.1E-02	2.9E-01	1.4E+00	3.5E+00	8.5E-06	5.1E-03	2.8E+00	1.7E+00	6.1E-06	3.2E+00	**5.1E-06**
F23	4	Best	**-1.1E+01**	-1.2E+00	-7.9E+00	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**
		Mean	-1.1E+01	-1.2E+00	-6.3E+00	-9.1E+00	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	-1.0E+01	**-1.1E+01**	**-1.1E+01**	-1.0E+01
		Std. dev	6.7E-02	9.1E-03	9.6E-01	2.6E+00	3.2E-05	5.8E-06	**6.2E-08**	6.3E-01	4.6E-06	5.4E-05	1.7E+00
F24	10	Best	2.0E-07	7.2E+00	1.6E-05	2.0E-07	2.0E-07	**2.0E-07**	**2.0E-07**	**2.0E-07**	**2.0E-07**	**2.0E-07**	**2.0E-07**
		Mean	**2.0E-07**	9.5E+00	1.7E-01	1.0E-02	**2.0E-07**	**2.0E-07**	7.9E-01	2.0E-01	4.0E-01	1.5E+00	2.0E-01
		Std. dev	1.0E+00	2.0E-01	6.4E-01	9.8E-01	1.0E+00	1.0E+00	**1.3E-01**	5.8E-01	1.6E-01	4.2E-01	3.7E-01
	30	Best	2.1E+02	1.2E+05	3.4E+04	1.3E+02	1.9E+02	5.6E-06	**0.0E+00**	4.4E-03	7.8E-04	1.3E-03	1.4E-02
		Mean	1.7E+03	1.2E+05	4.2E+04	7.7E+03	2.6E+03	7.3E-01	**2.9E-15**	4.4E+00	7.1E-01	1.6E+00	3.5E+00
		Std. dev	7.7E+02	6.9E+02	9.9E+03	1.4E+04	3.9E+03	1.2E+00	**2.0E-15**	5.5E+00	1.6E+00	2.7E+00	5.9E+00
	60	Best	1.8E+04	1.5E+06	3.1E+05	5.8E+03	1.8E+04	**4.2E-06**	5.3E-04	3.8E-03	7.4E-02	1.7E-02	1.5E-02
		Mean	2.1E+04	1.5E+06	4.9E+05	3.3E+05	7.4E+04	1.0E+01	**1.2E+00**	1.4E+01	1.0E+01	1.2E+01	1.9E+01
		Std. dev	1.6E+03	**1.6E-02**	8.2E+04	6.4E+05	8.1E+04	1.0E+01	3.3E+00	1.8E+01	1.3E+01	2.0E+01	2.1E+01
	90	Best	5.6E+04	3.6E+06	1.4E+06	1.2E+04	1.4E+05	5.1E-03	**4.2E-05**	3.2E-01	3.3E-03	4.5E-01	1.4E-03
		Mean	6.7E+04	3.6E+06	1.9E+06	1.4E+06	3.8E+05	**4.5E+00**	1.8E+02	2.6E+01	3.0E+01	4.9E+01	4.8E+01
		Std. dev	6.2E+03	1.0E+02	3.5E+05	2.8E+06	3.2E+05	**1.1E+01**	5.7E+02	3.0E+01	2.8E+01	4.7E+01	7.6E+01
F25	10	Best	1.6E+00	3.8E+00	1.4E+00	3.7E+00	4.2E+00	1.4E+00	9.7E-01	**9.7E-01**	1.0E+00	9.7E-01	9.7E-01
		Mean	2.5E+00	3.9E+00	2.5E+00	5.7E+00	4.2E+00	1.5E+00	1.4E+00	1.5E+00	1.3E+00	**1.3E+00**	1.4E+00
		Std. dev	7.4E-01	3.2E-02	7.5E-01	7.8E-01	**3.1E-04**	8.3E-02	2.7E-01	2.3E-01	1.9E-01	1.7E-01	2.0E-01

TABLE 10. ***(Continued.)*** **The best, mean, and standard deviation of test function values found by the ABC, BAT, CS, PSO, HS, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, AND E6GOA on the test optimization functions with (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.**

	30	Best	9.7E+03	4.5E+05	1.1E+05	5.2E+00	1.1E+02	2.5E-01	7.7E-01	2.5E-01	**2.5E-01**	9.6E-01	2.5E-01
		Mean	8.6E+04	1.3E+06	1.6E+05	8.4E+03	8.8E+03	3.9E-01	8.9E-01	**3.6E-01**	5.0E-01	1.1E+00	6.1E-01
		Std. dev	8.0E+04	2.8E+05	3.7E+04	2.3E+04	2.1E+04	2.3E-01	**7.9E-02**	3.3E-01	3.4E-01	1.3E-01	4.3E-01
	60	Best	1.4E+06	2.9E+06	8.3E+05	3.8E+02	6.8E+03	**2.5E-01**	9.2E-01	2.6E-01	2.6E-01	9.9E-01	2.6E-01
		Mean	2.6E+06	5.7E+06	1.4E+06	1.3E+06	1.3E+05	**6.7E-01**	9.5E-01	8.8E-01	1.0E+00	1.2E+00	8.6E-01
		Std. dev	8.2E+05	9.9E+05	3.2E+05	3.4E+06	2.5E+05	5.8E-01	**2.2E-02**	3.3E-01	4.9E-01	4.7E-01	4.0E-01
	90	Best	9.3E+06	1.1E+07	2.7E+06	1.2E+03	1.5E+05	**2.5E-01**	9.7E-01	2.5E-01	2.6E-01	1.0E+00	2.5E-01
		Mean	1.3E+07	1.1E+07	4.2E+06	4.5E+05	7.5E+05	8.8E-01	9.8E-01	1.2E+00	1.1E+00	1.6E+00	**8.2E-01**
		Std. dev	1.8E+06	1.3E+03	1.2E+06	9.9E+05	1.0E+06	5.1E-01	**7.4E-03**	9.6E-01	4.2E-01	8.4E-01	9.0E-01
F26	10	Best	3.0E-05	6.3E-04	3.0E-05	3.0E-05	5.0E-05	**3.0E-05**	**3.0E-05**	**3.0E-05**	**3.0E-05**	**3.0E-05**	**3.0E-05**
		Mean	3.0E-05	6.0E-02	**2.0E-05**	4.0E-05	3.8E-04	3.0E-05	3.0E-05	3.0E-05	3.0E-05	3.0E-05	3.0E-05
		Std. dev	1.0E+00	1.1E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	1.0E+00	**1.0E+00**	1.0E+00	1.0E+00
	30	Best	1.6E+01	1.2E+02	6.6E+01	5.1E-01	3.7E-01	3.3E-06	5.9E-01	**1.2E-07**	1.0E-06	6.6E-01	1.7E-06
		Mean	2.9E+01	1.2E+02	7.2E+01	1.8E+01	3.1E+00	5.7E-03	1.0E+00	5.0E-03	**2.8E-03**	1.2E+00	6.9E-03
		Std. dev	8.8E+00	**6.6E-05**	4.6E+00	2.8E+01	5.1E+00	8.1E-03	2.9E-01	5.4E-03	5.2E-03	2.5E-01	1.0E-02
	60	Best	1.8E+02	3.8E+02	1.8E+02	4.6E+00	7.2E+00	9.8E-06	1.9E+00	5.7E-06	**8.9E-08**	2.9E+00	5.1E-06
		Mean	2.2E+02	3.8E+02	2.2E+02	4.6E+01	2.4E+01	**2.9E-03**	2.6E+00	6.4E-03	3.5E-03	3.6E+00	6.3E-03
		Std. dev	2.8E+01	**1.1E-03**	2.5E+01	6.1E+01	2.6E+01	5.0E-03	5.5E-01	6.0E-03	5.2E-03	4.7E-01	9.0E-03
	90	Best	4.5E+02	7.9E+02	2.5E+02	7.3E+00	3.1E+01	1.8E-04	3.7E+00	**1.8E-08**	6.6E-06	5.6E+00	2.6E-05
		Mean	5.3E+02	9.2E+02	3.4E+02	1.2E+02	6.6E+01	1.7E-02	4.3E+00	8.6E-03	7.7E-03	6.3E+00	**6.7E-03**
		Std. dev	4.3E+01	4.5E+01	4.4E+01	1.7E+02	4.9E+01	2.0E-02	5.3E-01	9.1E-03	1.4E-02	4.3E-01	**7.9E-03**
F27	10	Best	**0.0E+00**	2.5E-03	4.0E-06	2.8E-02	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**
		Mean	1.0E-06	2.5E-03	1.4E-04	3.1E-01	3.4E-05	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**
		Std. dev	4.0E-01	4.0E-01	4.0E-01	**1.7E-01**	4.0E-01	4.0E-01	4.0E-01	4.0E-01	4.0E-01	4.0E-01	4.0E-01
	30	Best	1.2E+02	1.9E+02	6.1E+01	8.8E+00	4.5E+01	5.3E+01	5.6E+01	5.2E+01	5.1E+01	8.4E+01	5.3E+01
		Mean	1.6E+02	2.0E+02	1.4E+02	3.4E+01	**3.7E+00**	5.1E+01	4.2E+01	5.1E+01	5.1E+01	1.2E+02	5.1E+01
		Std. dev	2.1E+02	2.0E+02	2.5E+02	2.5E+02	2.5E+02	2.0E+02	2.4E+02	2.0E+02	**2.0E+02**	2.2E+02	2.0E+02
	60	Best	3.4E+02	3.1E+02	2.9E+02	3.6E+01	2.2E+02	3.3E+02	3.3E+02	3.3E+02	3.3E+02	1.3E+02	3.3E+02
		Mean	4.0E+02	3.1E+02	3.4E+02	1.1E+02	**1.1E+02**	3.3E+02	2.0E+02	3.3E+02	3.3E+02	1.5E+02	3.3E+02
		Std. dev	2.4E+02	2.0E+02	2.3E+02	2.8E+02	3.2E+02	2.0E+02	4.0E+02	**2.0E+02**	2.0E+02	2.1E+02	2.0E+02
	90	Best	6.2E+02	1.9E+02	4.5E+02	1.3E+02	2.8E+02	6.1E+02	6.1E+02	6.1E+02	6.1E+02	**1.1E+02**	6.1E+02
		Mean	6.7E+02	1.5E+02	5.1E+02	**1.5E+01**	1.3E+02	6.1E+02	3.4E+02	6.1E+02	6.1E+02	1.4E+02	6.1E+02
		Std. dev	2.3E+02	2.1E+02	2.3E+02	3.1E+02	3.7E+02	**2.0E+02**	5.5E+02	2.0E+02	2.0E+02	2.2E+02	2.0E+02
F28	10	Best	**0.0E+00**	**0.0E+00**	**0.0E+00**	4.8E+00	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**
		Mean	**0.0E+00**	**0.0E+00**	**0.0E+00**	2.3E+01	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**
		Std. dev	3.0E+00	3.0E+00	3.0E+00	5.0E+00	**3.0E+00**	3.0E+00	3.0E+00	3.0E+00	3.0E+00	3.0E+00	3.0E+00
	30	Best	2.2E+02	5.5E+03	1.1E+03	8.8E+00	1.3E+01	2.8E-06	**0.0E+00**	3.6E-05	2.2E-04	2.7E-05	3.1E-06
		Mean	4.8E+02	5.5E+03	1.6E+03	2.7E+02	1.9E+02	4.3E-02	1.5E-01	1.4E-02	2.6E-02	**3.7E-03**	1.3E-02
		Std. dev	1.4E+02	2.5E-01	4.4E+02	4.9E+02	3.6E+02	7.6E-02	4.9E-01	2.6E-02	4.4E-02	**7.7E-03**	2.1E-02
	60	Best	4.1E+03	1.7E+04	5.1E+03	2.0E+02	3.7E+02	4.8E-07	**0.0E+00**	6.9E-06	4.1E-05	6.0E-07	3.1E-06
		Mean	8.4E+03	1.7E+04	8.5E+03	1.5E+03	2.5E+03	4.0E-02	4.8E-02	6.5E-02	8.1E-02	**1.4E-02**	7.1E-02
		Std. dev	3.0E+03	2.8E-01	2.2E+03	1.6E+03	3.7E+03	5.2E-02	1.5E-01	2.0E-01	1.0E-01	**2.5E-02**	1.5E-01
	90	Best	3.1E+04	8.4E+04	7.0E+03	1.6E+02	1.7E+03	8.2E-06	**0.0E+00**	1.2E-05	7.8E-04	2.6E-04	7.8E-04
		Mean	3.8E+04	9.3E+04	1.4E+04	1.6E+03	6.6E+03	1.6E-01	**2.6E-17**	5.0E-02	1.2E-01	5.3E-02	1.5E-01
		Std. dev	4.7E+03	3.0E+03	5.0E+03	2.2E+03	7.4E+03	3.7E-01	**2.0E-17**	1.1E-01	1.5E-01	1.4E-01	2.4E-01
F29	10	Best	3.8E+00	2.0E+00	3.8E+00	**0.0E+00**	6.0E-05	2.0E+00	3.8E+00	2.0E+00	2.0E+00	3.8E+00	2.0E+00
		Mean	3.8E+00	2.0E+00	3.8E+00	2.3E-03	**1.3E-03**	2.0E+00	3.8E+00	2.0E+00	2.0E+00	3.8E+00	2.0E+00
		Std. dev	**3.9E+00**	3.9E+00	**3.9E+00**	3.9E+00	3.9E+00	3.9E+00	**3.9E+00**	3.9E+00	3.9E+00	**3.9E+00**	3.9E+00
	30	Best	6.7E+00	2.8E+00	2.7E+00	1.4E-01	1.9E+01	5.1E+00	3.3E+00	8.5E+00	7.1E+00	3.3E+00	6.4E+00
		Mean	4.9E+00	3.5E+00	**1.5E+00**	1.8E+00	1.7E+01	2.2E+00	3.2E+00	2.3E+00	3.7E+00	1.7E+00	2.6E+00
		Std. dev	1.1E+01	9.9E+00	1.0E+01	1.1E+01	1.2E+01	1.1E+01	**9.9E+00**	1.2E+01	1.1E+01	1.1E+01	1.1E+01
	60	Best	9.9E+00	2.4E+00	7.7E+00	7.1E+00	4.3E+01	1.8E+01	1.5E+01	1.7E+01	1.4E+01	1.1E+01	1.9E+01
		Mean	8.8E+00	**2.4E+00**	7.1E+00	4.6E+00	3.7E+01	1.1E+01	1.4E+01	1.2E+01	1.0E+01	8.0E+00	1.1E+01
		Std. dev	1.0E+01	**9.7E+00**	1.0E+01	1.1E+01	1.7E+01	1.4E+01	1.0E+01	1.4E+01	1.2E+01	1.1E+01	1.4E+01
	90	Best	1.6E+01	7.4E+00	1.3E+01	1.3E+01	6.7E+01	3.0E+01	2.4E+01	2.7E+01	2.9E+01	1.7E+01	2.7E+01
		Mean	1.3E+01	**6.3E+00**	1.2E+01	9.9E+00	5.8E+01	2.4E+01	1.8E+01	2.4E+01	2.3E+01	1.3E+01	2.3E+01
		Std. dev	1.2E+01	**1.0E+01**	1.1E+01	1.1E+01	2.2E+01	1.3E+01	1.4E+01	1.2E+01	1.4E+01	1.2E+01	1.2E+01
F30	10	Best	2.0E+00	2.2E+00	3.3E+00	1.5E-01	6.1E-04	2.2E+00	3.3E+00	2.2E+00	2.2E+00	3.3E+00	2.2E+00
		Mean	2.0E+00	2.2E+00	3.3E+00	3.4E-01	**2.2E-02**	2.2E+00	3.3E+00	2.2E+00	2.2E+00	3.3E+00	2.2E+00
		Std. dev	**3.3E+00**	3.3E+00	**3.3E+00**	3.4E+00	3.4E+00	3.3E+00	**3.3E+00**	3.3E+00	3.3E+00	**3.3E+00**	3.3E+00
	30	Best	2.7E+02	5.6E+02	3.0E+02	5.9E+01	2.4E+02	3.4E-10	**0.0E+00**	8.5E-04	1.5E-02	3.1E-02	2.6E-05
		Mean	3.4E+02	5.7E+02	3.9E+02	4.2E+08	2.9E+02	7.0E-02	**2.3E-04**	8.6E-01	4.7E-01	1.4E+00	5.7E+01
		Std. dev	4.2E+01	2.3E+00	5.4E+01	8.5E+08	5.1E+01	1.5E-01	**7.4E-04**	1.2E+00	7.1E-01	1.3E+00	5.5E+01
	60	Best	8.5E+02	1.2E+03	1.0E+03	1.2E+10	8.7E+02	6.2E-04	**0.0E+00**	8.0E-01	4.8E-03	4.7E-02	6.1E-01
		Mean	9.3E+02	1.4E+04	1.2E+03	7.2E+12	8.8E+02	8.2E+01	**5.9E+00**	1.6E+02	3.6E+02	2.4E+02	6.1E+02
		Std. dev	4.8E+01	4.6E+03	1.2E+02	7.0E+12	5.5E+01	2.4E+02	**9.2E+00**	3.1E+02	4.6E+02	3.4E+02	4.5E+02
	90	Best	1.3E+03	1.9E+03	2.1E+03	1.4E+13	1.3E+03	8.9E-04	**0.0E+00**	3.2E+00	9.0E+00	2.0E-02	2.6E+00
		Mean	1.7E+03	1.9E+04	6.1E+09	3.6E+14	1.4E+03	2.6E+02	**1.3E-17**	5.8E+02	5.3E+02	1.4E+02	8.4E+02

TABLE 10. *(Continued.)* **The best, mean, and standard deviation of test function values found by the ABC, BAT, CS, PSO, HS, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, AND E6GOA on the test optimization functions with (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.**

		Std. dev	2.1E+02	6.1E+03	5.1E+09	2.5E+14	1.1E+02	5.4E+02	**7.3E-18**	7.2E+02	8.1E+02	2.1E+02	8.8E+02
F31	10	Best	6.6E-03	9.2E+00	1.6E+00	6.8E-02	7.5E+00	**0.0E+00**	5.1E+00	**0.0E+00**	**0.0E+00**	5.1E+00	**0.0E+00**
		Mean	1.4E-01	9.6E+00	3.5E+00	1.4E+00	7.5E+00	**0.0E+00**	5.1E+00	**0.0E+00**	**0.0E+00**	5.1E+00	**0.0E+00**
		Std. dev	1.0E+01	1.0E+01	1.2E+01	1.2E+01	1.0E+01	1.0E+01	**1.0E+01**	1.0E+01	1.0E+01	1.0E+01	1.0E+01
	30	Best	4.9E+09	4.7E+10	1.0E+10	3.0E+07	6.6E+07	5.4E+01	**0.0E+00**	1.8E-03	1.4E+03	5.5E+01	8.0E+02
		Mean	1.0E+10	4.7E+10	1.0E+10	3.2E+09	1.3E+09	3.8E+05	5.1E+05	9.3E+05	3.3E+05	**1.3E+05**	8.7E+05
		Std. dev	4.4E+09	4.9E+07	**0.0E+00**	5.5E+09	2.4E+09	4.3E+05	1.5E+06	1.4E+06	4.3E+05	2.4E+05	1.4E+06
	60	Best	4.4E+10	5.6E+10	1.0E+10	6.2E+08	2.2E+09	1.6E+01	**1.5E+01**	9.1E+02	7.7E+02	6.7E+03	1.4E+04
		Mean	6.4E+10	5.6E+10	1.0E+10	1.0E+10	7.8E+09	6.0E+05	**1.9E+04**	6.3E+05	1.3E+06	4.5E+05	6.3E+05
		Std. dev	1.4E+10	1.8E+01	**0.0E+00**	1.4E+10	8.1E+09	8.5E+05	2.1E+04	4.9E+05	1.7E+06	6.3E+05	6.8E+05
	90	Best	1.1E+11	1.3E+11	1.0E+10	9.9E+08	9.1E+09	1.4E+00	**0.0E+00**	6.0E+02	1.4E+04	3.8E+01	2.0E+05
		Mean	1.3E+11	2.1E+11	1.0E+10	1.9E+10	2.0E+10	3.0E+05	**7.2E-10**	1.0E+06	7.9E+05	4.8E+05	1.1E+06
		Std. dev	1.3E+10	2.8E+10	**0.0E+00**	2.5E+10	1.4E+10	3.7E+05	3.5E-10	9.2E+05	8.2E+05	4.6E+05	9.7E+05
F32	10	Best	5.1E-03	8.3E+00	1.2E+00	9.8E-03	1.0E-04	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**
		Mean	3.8E-02	9.1E+00	3.3E+00	3.1E+00	1.0E-04	1.6E-03	1.3E+00	5.3E-01	**0.0E+00**	2.0E+00	**0.0E+00**
		Std. dev	1.1E+01	1.1E+01	1.2E+01	1.4E+01	1.0E+01	1.0E+01	1.3E+01	1.2E+01	1.0E+01	1.4E+01	**1.0E+01**
	30	Best	9.3E+02	5.3E+02	7.6E+02	1.1E+03	1.1E+03	1.1E+03	9.4E+02	1.1E+03	1.1E+03	9.3E+02	1.1E+03
		Mean	8.8E+02	**5.3E+02**	7.0E+02	9.4E+02	1.1E+03	1.1E+03	9.3E+02	1.1E+03	1.1E+03	8.7E+02	1.1E+03
		Std. dev	6.7E+01	**3.9E+01**	7.1E+01	2.7E+02	8.1E+01	3.9E+01	4.8E+01	3.9E+01	3.9E+01	6.8E+01	3.9E+01
	60	Best	1.6E+03	1.1E+03	1.3E+03	2.3E+03	2.3E+03	2.3E+03	1.9E+03	2.3E+03	2.3E+03	1.6E+03	2.3E+03
		Mean	1.5E+03	**1.1E+03**	1.2E+03	1.8E+03	2.2E+03	2.3E+03	1.8E+03	2.3E+03	2.3E+03	1.5E+03	2.3E+03
		Std. dev	8.2E+01	4.0E+01	8.2E+01	5.3E+02	2.1E+02	3.9E+01	1.6E+02	**3.9E+01**	3.9E+01	8.5E+01	3.9E+01
	90	Best	2.0E+03	1.5E+03	1.7E+03	3.4E+03	3.3E+03	3.5E+03	2.5E+03	3.5E+03	3.5E+03	2.1E+03	3.5E+03
		Mean	1.8E+03	**1.5E+03**	1.7E+03	2.6E+03	3.1E+03	3.5E+03	2.4E+03	3.5E+03	3.5E+03	2.1E+03	3.5E+03
		Std. dev	1.7E+02	**3.9E+01**	1.0E+02	8.5E+02	3.2E+02	3.9E+01	2.1E+02	3.9E+01	3.9E+01	8.8E+01	3.9E+01
F33	10	Best	7.7E-03	9.4E+00	2.6E+00	1.8E-03	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**
		Mean	6.3E-02	9.4E+00	4.3E+00	1.5E+00	**0.0E+00**	**0.0E+00**	**0.0E+00**	2.0E-01	**0.0E+00**	**0.0E+00**	5.4E-01
		Std. dev	1.1E+01	1.1E+01	1.2E+01	1.3E+01	1.1E+01	1.1E+01	**1.1E+01**	1.1E+01	1.1E+01	1.1E+01	1.2E+01
	30	Best	6.1E-04	2.9E-01	1.6E-02	1.0E-06	2.2E-13	5.8E-14	**0.0E+00**	4.1E-13	1.5E-11	2.0E-11	3.3E-12
		Mean	5.8E-01	3.0E-01	4.1E-02	9.6E-02	**5.9E-10**	3.8E-08	5.4E-08	7.7E-08	7.3E-08	1.7E-07	3.9E-07
		Std. dev	5.6E-01	3.3E-04	2.0E-02	2.2E-01	**1.9E-09**	9.7E-08	1.7E-07	1.8E-07	1.6E-07	2.6E-07	1.2E-06
	60	Best	5.9E+00	2.5E-01	5.1E-02	2.5E-03	9.9E-10	3.6E-12	**0.0E+00**	2.0E-11	2.2E-12	9.0E-13	2.0E-12
		Mean	9.5E+00	3.9E-01	1.6E-01	5.8E-01	8.3E-05	8.4E-09	7.8E-09	2.0E-07	1.8E-08	**6.1E-09**	5.4E-08
		Std. dev	1.9E+00	4.9E-02	7.3E-02	7.7E-01	2.6E-04	1.7E-08	1.7E-08	5.5E-07	2.5E-08	**1.2E-08**	7.1E-08
	90	Best	1.7E+01	3.7E-01	1.0E-01	1.2E-05	2.3E-08	1.6E-13	2.0E-14	3.5E-14	3.3E-11	**7.6E-16**	3.6E-13
		Mean	2.1E+01	4.8E-01	4.4E-01	6.4E-01	1.9E-03	7.2E-09	5.6E-08	4.9E-08	1.3E-07	**7.6E-10**	2.8E-09
		Std. dev	1.8E+00	3.8E-02	2.7E-01	6.5E-01	5.9E-03	1.7E-08	7.7E-08	6.7E-08	3.0E-07	**2.3E-09**	4.6E-09
F34	10	Best	1.8E-03	2.6E+03	1.1E+02	7.0E-01	4.7E-02	1.0E-08	**0.0E+00**	5.0E-06	2.4E-04	2.3E-04	4.6E-06
		Mean	3.2E-02	2.6E+03	2.2E+02	9.2E+01	6.0E+00	1.3E-02	8.3E-03	6.2E-02	5.6E-03	**1.5E-03**	5.8E-02
		Std. dev	3.3E-02	3.7E-03	9.2E+01	1.9E+02	1.8E+01	3.3E-02	1.9E-02	7.1E-02	1.0E-02	**1.5E-03**	1.2E-01
	30	Best	5.0E+04	2.9E+05	6.7E+04	5.0E+02	4.2E+02	6.2E-05	**3.5E-05**	3.4E-03	2.3E-02	7.3E-03	5.3E-03
		Mean	7.8E+04	2.9E+05	9.3E+04	1.8E+04	5.2E+03	1.4E+00	**4.6E-01**	2.9E+00	3.9E+00	1.5E+00	1.4E+00
		Std. dev	2.1E+04	**1.7E-02**	1.7E+04	2.6E+04	9.5E+03	1.9E+00	8.9E-01	5.8E+00	5.0E+00	2.1E+00	1.5E+00
	60	Best	6.8E+05	9.7E+05	3.9E+05	8.1E+03	3.1E+04	6.0E-04	**0.0E+00**	2.6E-01	6.2E-02	1.3E-01	3.2E-02
		Mean	9.6E+05	1.7E+06	5.4E+05	3.3E+05	1.0E+05	**4.2E+00**	6.2E+00	1.4E+01	8.6E+00	9.2E+00	1.1E+01
		Std. dev	1.2E+05	2.4E+05	9.7E+04	6.5E+05	1.1E+05	9.2E+00	**8.0E+00**	1.3E+01	1.3E+01	1.0E+01	8.3E+00
	90	Best	2.5E+06	4.2E+06	1.1E+06	1.9E+04	1.6E+05	2.3E-04	**0.0E+00**	1.9E-02	8.6E-01	2.3E-03	5.4E-01
		Mean	2.8E+06	4.2E+06	1.5E+06	8.0E+05	3.3E+05	**7.9E+00**	2.6E+01	1.4E+01	1.7E+01	1.8E+01	2.6E+01
		Std. dev	1.8E+05	2.4E+04	2.4E+05	1.6E+06	2.6E+05	**1.3E+01**	7.3E+01	1.3E+01	2.1E+01	2.6E+01	2.5E+01

Figure 7 shows the results obtained from 11 sample convergence plots when applying ABC, BAT, CS, HS, PSO, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA to solve the following test optimization functions (chosen randomly from Table 4: Step, Sphere, Schwefel 1.2, and Penalized No.2. As previously mentioned, these plots show the progress of the optimization process for each algorithm throughout the iterations by plotting the best achieved optimum per iteration. From these figures, it can be claimed that the six proposed algorithms are significantly superior to the other new metaheuristic algorithms used in this experiment over the optimization process in terms of both convergence speed and final result.

Figure 7(a) shows the results obtained for F6 Step function with 10 dimension. By carefully looking at this figure, we conclude that the E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA has a stably faster convergence speed than the other algorithms during the whole optimization process. The ABC, HS, and PSO algorithms demonstrated the second-best performance and outperform other optimization algorithms in terms of final global minimum and convergence speed in F6 function.

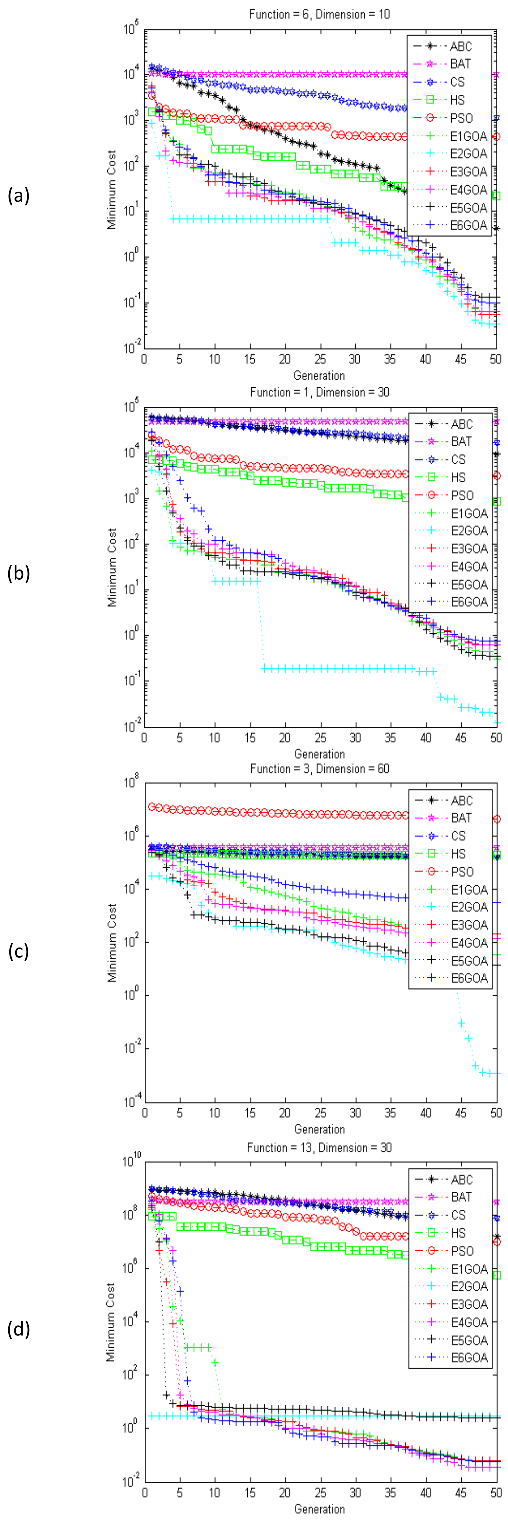

FIGURE 7. **Convergence curves of GOA, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA against 4 functions (A-F6, B- F1, C-F3, D-F13) with 10, 30, and 60 dimensions over 50 generations.**

The convergence trends of the test optimization function with 30 dimensions are depicted in Figure 7(b). It can be disclosed from these figures that the six proposed algorithms have fast convergence compared with other optimization algorithms. The convergence trends of the test optimization function with 60 dimension are depicted in Figure 7(c). It can be disclosed from this figure that the six proposed algorithms have fast convergence compared with other optimization algorithms. The E2GOA converges faster than E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA. Figure 7(d) shows the optimization process against (F13: Dim = 30), the E2GOA converges faster than E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA. ABC, BAT, CS, HS, and PSO algorithms fall into the trap of the local optimum and fail to find the best solution.

C. SCENARIO 3: PERFORMANCE OF EGOA AGAINST NEW OPTIMIZATION ALGORITHMS

In the past few years, many optimization algorithms have been proposed. There are many new optimization algorithms showing a clear and superior performance over the old proposed algorithms in the past decade. Consequently, it is important to compare the six proposed algorithms with a sufficient sample of the latest algorithms. Accordingly, we have selected six new algorithms published in high-quality journals, namely, ALO, DA, MBO, MFO, SCA, and SSA. The scenario that was implemented here is the same as in the previous scenarios.

In the experimental test of scenario 3, 34 well-known benchmark functions listed in Table 4 were used to verify the effectiveness of the six proposals for an enhanced GOA against new optimization algorithms, which are ALO, DA, MBO, MFO, SCA, and SSA. The benchmarks are divided into 2 types that can evaluate the different capabilities of the algorithms: Benchmarks F1-F13 and F24-F34 of the low-dimensional optimization functions that are set to 2, 3,4, and 6, Benchmarks F14-F23 of the high-dimensional optimization functions that are set to 10, 30, 60, and 90 listed in Table 11 similar to the previous experiments (Sec. V.B). In this experiment, the parameters for the six proposed algorithms are used the same as in the previous experiments. For ALO, DA, MBO, MFO, SCA, and SSA, we set the parameters based on the original works as follows. ALO, DA, MFO, SCA, and SSA algorithms are based on the implementation by Mirjalili (http://www.alimirjalili.com/ Projects.html). For MBO algorithm, the butterfly adjusting rate BAR = 5/12, migration Peri = 1.2, maximum step Smax = 1.0, keep = 2 and migration ratio $\rho = 5/12$.

Table 11 presents the best possible optimum solution obtained by various EGOAs and the new metaheuristic algorithms on 2, 3, 4, and 6 of the low dimensional optimization functions. It can be seen from the table that the E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA performance is slightly less superior to other algorithms, except the E2GOA algorithm that shows better statistical results compared to the other algorithms on the low dimensional optimization functions. The E2GOA algorithm also shows the best performance, especially in terms of the quality of the reached optimum value in most of the cases.

TABLE 11. The best, mean, and standard deviation of test function values found by the ALO, DA, MBO, MFO, SCA, SSA, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA on the test optimization functions with (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.

F	D	Statistics	ALO	DA	MBO	MFO	SCA	SSA	E1GOA	E2GOA	E3GOA	E4GOA	E5GOA	E6GOA
F1	10	Best	8.0E-01	**0.0E+00**	3.5E-09	1.2E+02	7.0E+00	2.6E-01	3.0E-04	**0.0E+00**	1.1E-02	2.9E-04	1.3E-04	5.2E-03
		Mean	1.1E+02	2.0E-01	2.5E-01	3.0E+02	1.2E+02	9.7E+00	5.9E-02	**4.3E-16**	9.1E-02	6.5E-02	5.4E-02	1.1E+02
		Std. dev	1.8E+02	5.8E-01	6.7E-01	1.3E+02	1.8E+02	1.1E+01	6.2E-02	**3.4E-16**	7.6E-02	1.1E-01	6.2E-02	3.4E-02
	30	Best	4.2E+03	**0.0E+00**	2.2E+03	2.1E+04	5.2E+03	1.2E+03	1.4E-08	**0.0E+00**	1.9E-05	1.6E-01	2.3E-06	2.2E-02
		Mean	6.2E+03	1.6E+02	3.0E+03	2.5E+04	1.1E+04	2.2E+03	6.4E-01	**9.2E-16**	4.0E-01	8.9E-01	4.1E-01	7.2E-01
		Std. dev	1.9E+03	5.0E+02	9.4E+02	3.0E+03	4.7E+03	6.5E+02	1.2E+00	**6.7E-16**	5.7E-01	7.3E-01	3.9E-01	7.0E-01
	60	Best	1.3E+04	**0.0E+00**	6.4E+03	9.4E+04	1.4E+04	5.6E+03	1.0E-03	5.8E-06	5.1E-04	5.3E-02	3.3E-02	8.4E-05
		Mean	1.9E+04	2.5E+03	7.5E+03	1.0E+05	3.9E+04	7.2E+03	**3.2E-01**	1.1E+00	4.6E-01	1.4E+00	7.8E-01	5.5E-01
		Std. dev	5.0E+03	4.7E+03	2.2E+03	5.2E+03	1.1E+04	8.2E+02	**4.4E-01**	1.5E+00	4.7E-01	1.5E+00	1.1E+00	9.1E-01
	90	Best	2.3E+04	6.1E+00	6.4E+01	1.8E+05	2.3E+04	8.4E+03	1.8E-05	**0.0E+00**	3.8E-05	2.8E-04	2.1E-06	3.8E-02
		Mean	3.2E+04	1.2E+03	6.3E+01	1.9E+05	7.6E+04	9.5E+03	**4.4E-01**	3.9E+00	1.8E+00	1.1E+00	5.3E-01	1.3E+00
		Std. dev	5.5E+03	2.4E+03	**0.0E+00**	6.2E+03	2.4E+04	5.9E+02	6.3E-01	5.0E+00	2.1E+00	1.4E+00	5.7E-01	1.2E+00
F2	10	Best	4.9E-02	**0.0E+00**	**0.0E+00**	3.4E+00	3.2E-01	1.7E-01	2.4E-04	**0.0E+00**	3.4E-03	6.4E-03	2.3E-03	8.7E-03
		Mean	5.8E+00	1.8E-01	**0.0E+00**	4.7E+00	8.3E-01	1.0E+00	5.8E-02	9.7E-03	3.2E-02	5.7E-02	5.2E-02	5.8E+00
		Std. dev	5.7E+00	4.6E-01	**0.0E+00**	1.1E+00	3.6E-01	6.2E-01	7.2E-02	1.6E-02	4.5E-02	5.7E-02	9.7E-02	1.2E-01
	30	Best	3.0E+01	6.5E-01	7.4E-01	1.1E+02	8.6E+00	1.8E+01	2.1E-03	**0.0E+00**	1.4E-02	2.2E-02	2.9E-03	1.5E-03
		Mean	3.8E+01	1.4E+00	4.1E+00	1.1E+02	2.2E+01	2.1E+01	1.9E-01	**1.3E-08**	3.0E-01	2.4E-01	1.1E-01	1.5E-01
		Std. dev	7.9E+00	1.7E+00	4.4E+00	1.1E+00	1.2E+01	1.9E+00	2.6E-01	**5.7E-09**	3.7E-01	2.3E-01	1.2E-01	2.2E-01
	60	Best	6.3E+01	8.8E+01	1.6E+02	2.5E+02	1.0E+01	4.9E+01	2.0E-03	**0.0E+00**	2.8E-02	8.4E-02	3.2E-02	6.9E-02
		Mean	7.3E+01	9.0E+01	1.6E+02	2.5E+02	7.8E+01	5.2E+01	4.0E-01	**1.8E-08**	4.6E-01	6.0E-01	4.4E-01	6.3E-01
		Std. dev	7.2E+00	5.4E+00	1.5E+00	1.2E-03	3.8E+01	2.6E+00	6.0E-01	**1.1E-08**	4.6E-01	6.0E-01	4.0E-01	6.3E-01
	90	Best	1.3E+02	1.0E+02	9.4E+01	4.0E+02	1.9E+01	8.3E+01	2.1E-03	**0.0E+00**	7.1E-02	1.3E-02	1.1E-02	1.1E-02
		Mean	1.6E+02	1.0E+02	9.4E+01	4.0E+02	5.9E+01	8.7E+01	4.5E-01	**2.5E-08**	4.8E-01	9.3E-01	3.5E-01	8.6E-01
		Std. dev	3.2E+01	3.7E+00	**1.5E-14**	3.1E-01	3.8E+01	3.3E+00	7.5E-01	1.2E-08	4.6E-01	1.1E+00	4.3E-01	1.3E+00
F3	10	Best	7.8E+01	0.0E+00	1.4E+03	6.0E+02	9.3E+01	1.1E+02	2.2E-07	0.0E+00	3.1E-02	9.7E-02	1.8E-02	2.6E-03
		Mean	7.5E+02	1.1E+01	2.0E+03	1.3E+03	4.7E+02	2.1E+02	1.2E-01	**2.1E-03**	5.5E-01	1.2E+00	7.2E-01	7.5E+02
		Std. dev	5.9E+02	2.9E+01	9.3E+02	5.5E+02	5.1E+02	6.1E+01	1.6E-01	**3.4E-03**	6.4E-01	1.2E+00	8.9E-01	5.2E+00
	30	Best	8.5E+03	2.4E+04	2.0E+04	2.9E+04	2.0E+04	2.3E+03	9.2E-05	**0.0E+00**	2.9E-01	8.2E-03	2.5E-01	5.6E-01
		Mean	1.4E+04	2.4E+04	2.1E+04	4.4E+04	3.5E+04	3.5E+03	9.5E+00	**1.3E-15**	1.3E+01	1.7E+01	9.2E+00	2.5E+02
		Std. dev	3.8E+03	1.5E+02	3.7E+02	7.5E+03	8.8E+03	1.1E+03	1.9E+01	**5.8E-16**	1.1E+01	3.2E+01	1.9E+01	2.7E+02
	60	Best	3.1E+04	6.9E+04	7.0E+04	1.4E+05	1.5E+05	1.6E+04	8.6E-05	**0.0E+00**	2.9E+00	2.4E-02	2.6E-02	1.9E-01
		Mean	6.6E+04	7.1E+04	7.0E+04	1.8E+05	1.8E+05	2.4E+04	6.6E+00	**5.9E-15**	2.1E+02	9.5E+01	7.5E+00	8.6E+02
		Std. dev	2.6E+04	4.2E+03	5.6E+02	1.8E+04	2.8E+04	5.4E+03	1.9E+01	**4.3E-15**	2.5E+02	1.6E+02	1.7E+01	1.5E+03
	90	Best	8.1E+04	1.3E+05	7.0E+05	3.4E+05	2.8E+05	4.1E+04	1.0E-04	**0.0E+00**	4.9E-01	1.0E+01	3.2E+00	2.2E+00
		Mean	1.5E+05	1.3E+05	7.4E+05	4.6E+05	4.4E+05	5.6E+04	3.5E+01	**9.0E-15**	3.4E+02	8.1E+01	2.0E+01	2.6E+04
		Std. dev	6.5E+04	1.3E+03	2.9E+04	8.5E+04	7.5E+04	1.3E+04	4.8E+01	**8.8E-15**	4.3E+02	8.2E+01	1.4E+01	3.3E+04
F4	10	Best	2.5E+00	**0.0E+00**	6.2E-03	1.1E+01	4.3E+00	2.2E+00	1.2E-02	**0.0E+00**	1.3E-02	4.9E-03	7.7E-03	6.0E-03
		Mean	7.5E+00	5.1E-01	2.6E-01	1.4E+01	1.4E+01	3.7E+00	1.3E-01	**2.1E-02**	1.3E-01	7.6E-02	1.2E-01	7.5E+00
		Std. dev	3.6E+00	1.3E+00	4.0E-01	2.6E+00	7.0E+00	1.6E+00	9.6E-02	6.7E-02	1.4E-01	**6.1E-02**	1.2E-01	8.4E-02
	30	Best	2.1E+01	4.0E+00	3.3E+01	6.4E+01	6.6E+01	1.5E+01	2.1E-03	**0.0E+00**	8.5E-03	1.1E-02	3.1E-03	1.6E-03
		Mean	2.8E+01	8.2E+00	4.5E+01	7.3E+01	7.5E+01	1.9E+01	1.5E-01	**7.0E-05**	1.4E-01	1.2E-01	1.8E-01	7.2E-02
		Std. dev	4.4E+00	6.7E+00	6.0E+00	6.2E+00	3.8E+00	2.6E+00	1.5E-01	**2.2E-04**	1.3E-01	9.1E-02	1.8E-01	7.9E-02
	60	Best	2.5E+01	1.9E+00	4.4E+00	8.6E+01	8.6E+01	1.6E+01	7.1E-04	**0.0E+00**	3.7E-02	2.4E-04	1.0E-02	1.6E-02
		Mean	3.2E+01	1.6E+01	4.5E+00	8.9E+01	9.2E+01	1.8E+01	**6.2E-02**	1.6E-01	1.9E-01	1.1E-01	1.7E-01	2.4E-01
		Std. dev	3.6E+00	1.2E+01	1.1E-01	1.2E+00	2.2E+00	1.4E+00	**9.1E-02**	4.4E-01	1.1E-01	9.9E-02	1.5E-01	2.2E-01
	90	Best	3.3E+01	6.2E+00	6.3E+01	9.3E+01	9.4E+01	2.2E+01	1.6E-03	**0.0E+00**	8.8E-03	1.1E-03	2.3E-02	4.3E-02
		Mean	3.7E+01	2.2E+01	6.3E+01	9.4E+01	9.5E+01	2.6E+01	**1.1E-01**	2.1E-01	2.2E-01	1.4E-01	1.6E-01	1.7E-01
		Std. dev	2.7E+00	1.5E+01	**7.0E-02**	8.9E-01	7.6E-01	1.6E+00	1.1E-01	2.3E-01	1.7E-01	8.3E-02	8.6E-02	1.6E-01
F5	10	Best	2.0E+02	8.7E+00	2.7E+01	1.4E+03	1.1E+03	6.1E+01	3.9E-02	8.3E+00	7.9E-03	1.9E-03	8.9E+00	**1.4E-03**
		Mean	5.7E+03	2.8E+01	1.7E+03	2.8E+04	6.7E+03	2.6E+02	**3.9E+00**	9.0E+00	4.2E+00	4.3E+00	9.5E+00	5.7E+03
		Std. dev	1.2E+04	5.3E+01	3.4E+03	3.5E+04	7.8E+03	2.2E+02	4.6E+00	**6.7E-01**	4.4E+00	5.0E+00	7.9E-01	4.6E+00
	30	Best	6.1E+05	3.0E+01	3.1E+01	2.9E+07	1.4E+07	1.7E+05	9.0E-02	2.9E+01	**2.6E-05**	2.3E-03	2.9E+01	5.5E-04
		Mean	1.6E+06	3.7E+04	3.8E+01	4.1E+07	3.0E+07	3.8E+05	1.9E+01	3.0E+01	1.3E+01	**5.8E+00**	3.3E+01	1.8E+01
		Std. dev	1.0E+06	1.0E+05	8.4E+00	1.0E+07	1.1E+07	1.7E+05	1.6E+01	**3.3E+00**	1.7E+01	9.3E+00	6.6E+00	1.9E+01
	60	Best	9.2E+06	5.9E+01	3.4E+08	3.0E+08	2.6E+08	7.7E+05	3.4E-02	5.9E+01	6.8E-03	4.0E-01	5.9E+01	**1.8E-03**
		Mean	1.6E+07	2.9E+06	5.2E+08	3.6E+08	3.5E+08	1.7E+06	3.4E+01	5.9E+01	2.6E+01	3.1E+01	6.9E+01	**2.3E+01**
		Std. dev	8.4E+06	8.5E+06	1.2E+08	2.6E+07	7.8E+07	5.1E+05	3.4E+01	**1.1E-01**	3.1E+01	3.2E+01	1.2E+01	3.2E+01
	90	Best	8.2E+06	3.4E+03	2.2E+02	7.2E+08	2.8E+08	3.5E+06	4.9E-01	8.9E+01	7.7E-02	**1.2E-02**	8.9E+01	1.9E-01
		Mean	1.9E+07	3.1E+06	2.2E+02	8.3E+08	4.4E+08	4.4E+06	3.0E+01	1.0E+02	6.2E+01	**3.0E+01**	9.7E+01	5.0E+01
		Std. dev	5.7E+06	7.6E+06	**3.0E-14**	6.3E+07	9.1E+07	6.0E+05	4.5E+01	3.0E+01	5.1E+01	4.1E+01	7.0E+00	4.8E+01
F6	10	Best	5.4E-01	7.9E-04	**3.1E-06**	1.8E+02	8.4E+00	5.7E-01	7.5E-04	3.1E-03	6.7E-05	3.5E-05	3.2E-02	6.8E-04
		Mean	1.1E+02	1.3E+01	9.4E-02	3.2E+02	7.8E+01	8.1E+00	1.0E-01	**3.2E-02**	7.6E-02	8.1E-02	1.4E-01	1.1E+02
		Std. dev	1.3E+02	3.8E+01	2.6E-01	1.0E+02	9.9E+01	9.8E+00	8.1E-02	**3.6E-02**	8.0E-02	8.5E-02	7.2E-02	1.3E-01
	30	Best	3.2E+03	8.5E-01	4.4E+01	1.6E+04	3.9E+03	1.3E+03	2.3E-03	7.1E-02	1.2E-03	**1.8E-07**	2.8E+00	9.5E-04
		Mean	5.4E+03	1.2E+02	1.8E+03	2.4E+04	1.1E+04	1.9E+03	5.1E-01	7.2E-01	**2.4E-01**	2.1E+00	4.7E+00	5.2E-01
		Std. dev	1.3E+03	3.1E+02	5.4E+03	4.6E+03	4.4E+03	5.5E+02	7.3E-01	9.8E-01	**3.5E-01**	3.5E+00	1.3E+00	5.4E-01
	60	Best	9.9E+03	8.1E+00	5.2E+04	8.0E+04	3.1E+04	6.6E+03	1.8E-03	6.0E+00	**2.4E-04**	7.9E-03	1.1E+01	3.6E-02

TABLE 11. ***(Continued.)*** **The best, mean, and standard deviation of test function values found by the ALO, DA, MBO, MFO, SCA, SSA, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA on the test optimization functions with (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.**

		Mean	1.5E+04	1.8E+02	5.2E+04	8.7E+04	5.2E+04	7.6E+03	**5.8E-01**	1.2E+01	1.2E+00	6.9E-01	1.2E+01	9.2E-01
		Std. dev	3.7E+03	4.6E+02	**7.7E-12**	4.0E+03	1.5E+04	8.0E+02	4.9E-01	6.0E+00	1.6E+00	5.0E-01	1.2E+00	1.1E+00
	90	Best	2.6E+04	7.0E+01	2.5E+04	1.8E+05	2.9E+04	1.2E+04	1.5E-02	1.0E+01	**2.6E-03**	3.1E-02	1.7E+01	5.1E-02
		Mean	3.1E+04	6.0E+02	2.9E+04	2.0E+05	9.1E+04	1.3E+04	1.2E+00	1.5E+01	1.3E+00	**7.4E-01**	1.9E+01	1.2E+00
		Std. dev	4.3E+03	5.2E+02	3.2E+03	8.2E+03	3.2E+04	8.3E+02	1.9E+00	4.3E+00	1.1E+00	**6.3E-01**	1.3E+00	8.3E-01
F7	10	Best	4.0E-02	3.9E-03	3.8E-02	1.8E-02	1.3E-03	2.0E-02	1.2E-03	4.8E-04	**4.3E-04**	5.6E-04	4.9E-04	7.1E-04
		Mean	1.2E-01	**6.4E-03**	5.3E-02	1.2E-01	1.7E-02	1.2E-01	6.7E-03	1.5E-02	1.3E-02	7.0E-03	1.7E-02	1.2E-01
		Std. dev	8.3E-02	**2.6E-03**	7.6E-03	6.7E-02	9.8E-03	9.8E-02	6.6E-03	9.4E-03	1.2E-02	7.2E-03	1.2E-02	1.5E-02
	30	Best	1.6E+00	2.4E-02	1.2E+01	1.4E+01	1.9E-02	3.8E+00	**3.3E-04**	6.5E-04	4.4E-04	8.7E-04	4.4E-03	1.0E-03
		Mean	3.8E+00	5.4E-01	1.4E+01	2.8E+01	2.7E+00	6.1E+00	**8.5E-03**	3.0E-02	1.5E-02	1.2E-02	2.5E-02	1.8E-02
		Std. dev	1.5E+00	1.2E+00	1.0E+00	9.8E+00	2.5E+00	1.4E+00	**9.1E-03**	2.6E-02	1.3E-02	1.3E-02	2.1E-02	1.5E-02
	60	Best	1.8E+01	2.4E-02	3.5E+02	3.6E+02	6.5E+00	3.3E+01	4.0E-04	2.0E-03	3.5E-04	2.3E-03	1.8E-04	**1.5E-04**
		Mean	3.1E+01	8.2E-01	3.5E+02	4.3E+02	4.0E+01	8.1E+01	6.0E-03	5.9E-02	**5.0E-03**	1.3E-02	1.4E-02	1.4E-02
		Std. dev	8.0E+00	1.9E+00	3.3E-01	3.8E+01	3.7E+01	2.7E+01	1.1E-02	1.0E-01	**8.2E-03**	1.9E-02	1.1E-02	1.8E-02
	90	Best	8.2E+01	2.7E-02	6.0E+02	1.3E+03	3.3E+01	2.0E+02	3.5E-04	1.2E-02	**1.9E-04**	2.6E-04	2.1E-04	1.7E-03
		Mean	1.1E+02	2.7E+00	6.0E+02	1.5E+03	1.3E+02	2.6E+02	1.3E-02	6.0E-02	**4.3E-03**	1.0E-02	2.1E-02	6.2E-03
		Std. dev	2.3E+01	5.4E+00	2.7E-01	1.6E+02	7.4E+01	3.4E+01	2.1E-02	4.9E-02	**4.7E-03**	7.8E-03	2.0E-02	5.7E-03
F8	10	Best	2.8E+03	2.3E+03	3.8E+03	3.2E+03	**1.9E+03**	3.3E+03	3.8E+03	2.7E+03	3.8E+03	3.8E+03	2.9E+03	3.8E+03
		Mean	-2.6E+03	-2.7E+03	-4.0E+03	-3.3E+03	-1.9E+03	-3.0E+03	-4.2E+03	-3.1E+03	-4.2E+03	**-4.2E+03**	-3.1E+03	-2.6E+03
		Std. dev	4.7E+02	2.0E+00	6.2E+02	1.7E+02	1.9E+02	4.2E+02	2.3E-01	**8.2E-02**	1.5E-01	1.2E-01	8.0E+01	1.5E-01
	30	Best	6.7E+03	7.0E+03	4.2E+03	6.7E+03	**3.4E+03**	6.4E+03	1.2E+04	7.6E+03	1.2E+04	1.2E+04	7.5E+03	1.2E+04
		Mean	-5.5E+03	-7.3E+03	-4.6E+03	-6.3E+03	-3.4E+03	-6.3E+03	-1.3E+04	-8.0E+03	-1.3E+04	-1.3E+04	-7.6E+03	**-1.3E+04**
		Std. dev	5.7E+02	2.2E+02	1.3E+02	5.0E+02	2.3E+02	2.6E+02	2.2E+00	7.3E+01	9.7E-01	2.7E+00	3.2E+02	**5.9E-01**
	60	Best	1.1E+04	1.1E+04	2.4E+04	1.0E+04	**5.1E+03**	9.9E+03	2.5E+04	1.4E+04	2.5E+04	2.5E+04	1.2E+04	2.5E+04
		Mean	-1.1E+04	-1.1E+04	-2.4E+04	-8.7E+03	-4.7E+03	-8.5E+03	-2.5E+04	-1.4E+04	-2.5E+04	-2.5E+04	-1.1E+04	**-2.5E+04**
		Std. dev	3.4E+01	4.6E+02	5.2E+01	8.6E+02	4.9E+02	1.1E+03	2.0E+00	6.4E+02	1.9E+00	4.0E+00	6.8E+02	**7.0E-01**
	90	Best	1.9E+04	1.1E+04	5.8E+03	1.1E+04	**5.8E+03**	1.2E+04	3.7E+04	1.7E+04	3.7E+04	3.7E+04	1.5E+04	3.7E+04
		Mean	-1.7E+04	-1.2E+04	-6.2E+03	-1.0E+04	-5.6E+03	-1.0E+04	**-3.8E+04**	-1.6E+04	-3.8E+04	-3.8E+04	-1.4E+04	-3.8E+04
		Std. dev	1.4E+03	4.6E+02	**9.6E-13**	1.0E+03	3.5E+02	9.4E+02	1.2E+00	1.1E+03	2.2E+00	5.8E+00	1.0E+03	3.4E+00
F9	10	Best	3.0E+01	1.0E+01	1.8E-03	1.6E+01	3.0E+01	1.0E+01	1.1E-09	**0.0E+00**	2.6E-05	1.2E-05	9.3E-05	1.9E-07
		Mean	3.0E+01	1.6E+01	3.0E+00	2.3E+01	3.0E+01	2.6E+01	3.9E-04	**0.0E+00**	1.0E+00	3.6E-01	2.2E-03	3.0E+01
		Std. dev	2.8E-03	5.4E+00	3.7E+00	4.5E+00	**0.0E+00**	8.3E+00	6.7E-04	**0.0E+00**	3.1E+00	1.1E+00	3.9E-03	3.1E+00
	30	Best	1.9E+02	1.9E+02	1.1E+00	1.8E+02	1.9E+02	1.9E+02	2.0E-06	**0.0E+00**	7.7E-05	5.4E-04	8.1E-05	3.7E-05
		Mean	1.9E+02	1.9E+02	1.7E+00	1.8E+02	1.9E+02	1.9E+02	3.0E+00	**0.0E+00**	9.7E-02	6.1E+00	1.0E-01	3.1E+00
		Std. dev	5.3E-02	6.9E-02	1.4E+00	4.9E+00	**0.0E+00**	6.7E-02	9.4E+00	**0.0E+00**	1.5E-01	1.3E+01	1.4E-01	9.4E+00
	60	Best	3.5E+02	3.6E+02	2.3E+02	3.3E+02	3.6E+02	3.6E+02	9.1E-06	**0.0E+00**	5.8E-04	8.5E-05	1.2E-04	1.7E-03
		Mean	3.6E+02	3.6E+02	2.3E+02	3.5E+02	3.6E+02	3.6E+02	8.1E-02	**2.2E-03**	6.1E+00	6.0E+00	2.5E-01	1.5E+01
		Std. dev	1.2E+00	5.9E-01	1.5E+00	9.0E+00	**0.0E+00**	1.1E+00	2.2E-01	2.4E-03	1.9E+01	1.9E+01	5.2E-01	2.5E+01
	90	Best	6.8E+02	4.7E+02	4.1E+02	6.7E+02	6.8E+02	6.8E+02	8.2E-05	**0.0E+00**	1.3E-04	3.6E-04	5.8E-04	3.2E-05
		Mean	6.8E+02	5.4E+02	4.1E+02	6.7E+02	6.8E+02	6.8E+02	4.6E+00	**0.0E+00**	9.1E+00	1.9E-01	8.8E-01	2.8E+01
		Std. dev	2.2E-01	9.7E+01	8.1E-01	4.2E+00	**0.0E+00**	1.8E-01	1.2E+01	**0.0E+00**	2.8E+01	2.8E-01	2.3E+00	4.2E+01
F10	10	Best	2.6E+00	8.0E-15	8.0E-04	6.7E+00	2.3E+00	2.2E+00	9.8E-04	**8.9E-16**	5.7E-04	2.9E-03	1.7E-04	3.7E-03
		Mean	5.4E+00	3.8E-01	4.9E-01	9.5E+00	3.9E+00	3.6E+00	1.9E-01	**9.8E-09**	2.7E-01	2.8E-01	9.0E-02	5.4E+00
		Std. dev	1.8E+00	9.8E-01	1.5E+00	3.0E+00	7.5E-01	9.8E-01	2.7E-01	**4.3E-09**	3.6E-01	4.3E-01	1.4E-01	1.5E-01
	30	Best	1.1E+01	1.3E+00	8.7E+00	1.8E+01	1.8E+01	8.1E+00	7.8E-04	**8.9E-16**	6.8E-03	1.9E-02	1.9E-03	1.4E-02
		Mean	1.4E+01	3.6E+00	9.0E+00	1.9E+01	1.9E+01	9.9E+00	**8.7E-02**	2.5E-01	3.5E-01	2.5E-01	1.5E-01	3.5E-01
		Std. dev	1.9E+00	3.7E+00	1.2E+00	**5.9E-02**	1.0E-01	1.1E+00	9.5E-02	4.3E-01	2.9E-01	2.1E-01	1.8E-01	3.4E-01
	60	Best	1.3E+01	1.9E+01	1.8E+01	1.9E+01	1.9E+01	1.2E+01	8.2E-04	**8.9E-16**	8.7E-04	1.3E-03	3.8E-04	1.2E-03
		Mean	1.4E+01	1.9E+01	1.8E+01	1.9E+01	1.9E+01	1.2E+01	**1.3E-01**	1.3E-01	2.1E-01	1.8E-01	2.3E-01	3.1E-01
		Std. dev	7.1E-01	1.4E-05	2.8E-02	2.8E-03	**0.0E+00**	3.0E-01	2.3E-01	1.4E-01	2.3E-01	2.1E-01	1.7E-01	1.9E-01
	90	Best	1.6E+01	1.6E+01	1.9E+01	1.9E+01	1.9E+01	1.2E+01	1.1E-03	**8.9E-16**	4.5E-03	1.7E-02	1.5E-03	2.6E-03
		Mean	1.9E+01	1.9E+01	1.9E+01	1.9E+01	1.9E+01	1.2E+01	7.5E-02	**2.2E-02**	2.5E-01	1.6E-01	7.2E-02	3.0E-01
		Std. dev	1.1E+00	1.3E+00	**3.7E-15**	4.2E-04	**3.7E-15**	3.4E-01	1.2E-01	6.9E-02	1.9E-01	1.4E-01	5.7E-02	1.4E-01
F11	10	Best	4.6E-01	3.5E-01	4.2E-01	2.1E+00	7.9E-01	2.7E-01	4.5E-05	**0.0E+00**	6.2E-03	7.8E-03	3.6E-04	8.4E-06
		Mean	1.9E+00	4.2E-01	5.7E-01	3.3E+00	1.3E+00	4.6E-01	2.7E-01	**1.4E-01**	1.8E-01	2.7E-01	2.0E-01	1.9E+00
		Std. dev	1.4E+00	1.9E-01	**1.3E-01**	1.3E+00	3.2E-01	2.0E-01	2.0E-01	2.3E-01	1.4E-01	2.0E-01	1.8E-01	1.6E-01
	30	Best	2.9E+01	**0.0E+00**	6.5E+00	8.8E+01	3.8E+01	1.4E+01	5.3E-05	**0.0E+00**	7.8E-03	2.7E-03	4.5E-03	4.7E-03
		Mean	4.5E+01	3.4E+00	1.0E+01	1.9E+02	8.1E+01	2.3E+01	1.9E-01	**3.8E-15**	5.1E-01	2.2E-01	2.9E-01	5.0E-01
		Std. dev	9.8E+00	8.9E+00	4.3E+00	5.0E+01	2.4E+01	4.2E+00	3.2E-01	**3.2E-15**	3.6E-01	2.6E-01	3.9E-01	3.2E-01
	60	Best	1.1E+02	1.3E+00	2.2E+01	7.5E+02	2.1E+02	5.6E+01	3.5E-05	**0.0E+00**	3.5E-02	1.6E-02	1.5E-06	3.2E-03
		Mean	1.5E+02	9.4E+00	3.8E+01	8.3E+02	4.4E+02	6.6E+01	4.0E-01	**2.5E-01**	5.1E-01	4.4E-01	4.0E-01	4.2E-01
		Std. dev	5.4E+01	1.4E+01	4.7E+01	3.6E+01	1.6E+02	5.2E+00	3.7E-01	4.3E-01	3.5E-01	2.5E-01	2.8E-01	**2.3E-01**
	90	Best	1.9E+02	**0.0E+00**	1.5E+03	1.7E+03	5.5E+02	9.4E+01	**0.0E+00**	**0.0E+00**	6.8E-03	8.3E-02	4.2E-04	3.8E-04
		Mean	2.8E+02	1.5E+01	1.5E+03	1.8E+03	8.6E+02	1.1E+02	1.9E-01	**1.0E-01**	5.1E-01	4.8E-01	2.9E-01	2.3E-01
		Std. dev	4.7E+01	2.7E+01	**2.4E-13**	5.8E+01	1.6E+02	1.1E+01	2.0E-01	1.7E-01	3.8E-01	3.3E-01	3.4E-01	2.8E-01
F12	10	Best	1.1E+00	1.1E-03	**4.8E-12**	6.2E+00	2.1E-01	1.5E+00	7.1E-04	8.2E-05	1.5E-05	5.7E-04	8.7E-03	1.2E-04
		Mean	9.0E+00	4.2E-01	1.3E-01	1.7E+01	3.6E+00	3.2E+00	9.8E-03	**1.6E-03**	4.5E-03	1.1E-02	8.0E-02	9.0E+00
		Std. dev	4.0E+00	1.1E+00	2.7E-01	1.1E+01	3.1E+00	1.3E+00	7.8E-03	**2.4E-03**	5.7E-03	2.0E-02	7.0E-02	6.9E-03

TABLE 11. *(Continued.)* **The best, mean, and standard deviation of test function values found by the ALO, DA, MBO, MFO, SCA, SSA, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA on the test optimization functions with (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.**

	30	Best	2.1E+01	1.5E-01	2.3E-05	3.1E+07	7.9E+06	1.0E+01	9.1E-05	7.6E-03	2.9E-06	**8.6E-07**	9.5E-02	9.7E-07
		Mean	3.1E+04	7.8E+00	8.1E-01	6.3E+07	6.1E+07	2.0E+01	**1.2E-03**	4.7E-02	2.7E-03	3.0E-03	1.7E-01	1.6E-03
		Std. dev	5.2E+04	1.9E+01	2.6E+00	2.0E+07	6.6E+07	1.3E+01	**1.1E-03**	4.8E-02	3.0E-03	3.0E-03	5.1E-02	1.6E-03
	60	Best	1.4E+05	3.9E+00	8.4E-04	5.0E+08	2.8E+08	2.4E+01	3.3E-05	7.7E-02	1.2E-05	6.8E-04	3.1E-01	**4.1E-06**
		Mean	7.1E+05	9.6E+00	2.6E+08	6.6E+08	5.7E+08	7.7E+01	**9.5E-04**	1.2E-01	1.2E-03	2.4E-03	4.3E-01	1.4E-03
		Std. dev	1.2E+06	1.3E+01	2.8E+08	9.3E+07	2.0E+08	9.2E+01	**1.1E-03**	6.2E-02	1.1E-03	1.7E-03	7.5E-02	1.5E-03
	90	Best	7.1E+05	3.8E-01	6.5E+07	1.2E+09	1.0E+09	3.3E+01	1.3E-05	2.0E-01	**2.7E-06**	6.6E-05	4.1E-01	2.4E-05
		Mean	3.5E+06	5.7E+05	6.7E+07	1.4E+09	1.2E+09	2.1E+03	1.8E-03	2.9E-01	**8.4E-04**	1.7E-03	5.7E-01	1.2E-03
		Std. dev	2.8E+06	1.8E+06	3.6E+06	1.3E+08	1.4E+08	2.3E+03	2.0E-03	7.9E-02	**8.9E-04**	2.7E-03	8.5E-02	1.3E-03
F13	10	Best	4.2E+00	6.3E-05	**2.9E-07**	1.7E+01	3.5E-01	1.5E-01	1.9E-03	1.3E-04	1.4E-04	2.3E-05	1.4E-02	8.1E-04
		Mean	1.9E+02	3.9E-02	6.8E-01	1.2E+04	7.1E+02	1.3E+00	2.5E-02	**2.4E-03**	1.8E-02	2.1E-02	3.7E-02	1.9E+02
		Std. dev	5.5E+02	1.1E-01	2.2E+00	2.0E+04	1.7E+03	1.4E+00	2.0E-02	**3.6E-03**	2.3E-02	1.8E-02	1.9E-02	2.2E-02
	30	Best	4.9E+04	1.6E+00	2.2E-01	9.0E+07	2.6E+07	4.2E+02	2.1E-04	3.0E+00	2.3E-03	3.6E-04	1.3E+00	**4.9E-05**
		Mean	1.1E+06	8.3E+03	8.1E-01	1.7E+08	1.2E+08	1.1E+04	3.9E-02	3.0E+00	3.7E-02	2.6E-02	2.2E+00	**1.6E-02**
		Std. dev	1.2E+06	2.6E+04	8.6E-01	5.5E+07	6.3E+07	1.8E+04	3.4E-02	**8.6E-03**	3.5E-02	2.2E-02	7.1E-01	1.6E-02
	60	Best	4.1E+06	5.5E+00	1.9E+07	7.8E+08	5.5E+08	3.7E+05	**9.7E-06**	6.0E+00	1.6E-03	3.3E-04	5.5E+00	9.7E-06
		Mean	1.2E+07	1.2E+06	2.0E+07	1.1E+09	1.3E+09	1.1E+06	**2.6E-02**	6.7E+00	4.4E-02	7.2E-02	6.3E+00	8.2E-02
		Std. dev	6.0E+06	3.6E+06	8.5E+05	1.5E+08	4.2E+08	4.9E+05	**3.2E-02**	1.1E+00	3.6E-02	7.0E-02	6.1E-01	8.7E-02
	90	Best	1.1E+07	1.1E+01	2.9E+09	2.7E+09	1.1E+09	1.4E+06	1.0E-04	9.0E+00	3.2E-04	**3.0E-05**	8.4E+00	4.1E-04
		Mean	3.8E+07	1.8E+06	3.1E+09	3.6E+09	2.4E+09	3.3E+06	3.8E-02	9.0E+00	7.4E-02	**2.7E-02**	1.0E+01	4.0E-02
		Std. dev	2.6E+07	5.4E+06	1.7E+08	3.6E+08	6.7E+08	9.9E+05	3.5E-02	**1.1E-02**	7.1E-02	3.5E-02	1.4E+00	4.3E-02
F14	2	Best	3.0E+00	**1.0E+00**	**1.0E+00**	**1.0E+00**	1.0E+00	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**	**1.0E+00**
		Mean	6.0E+00	3.9E+00	1.7E+00	1.6E+00	3.4E+00	4.2E+00	**1.0E+00**	1.8E+00	1.2E+00	1.4E+00	2.5E+00	1.2E+00
		Std. dev	4.1E+00	3.4E+00	1.2E+00	1.3E+00	2.8E+00	3.1E+00	**2.2E-10**	1.1E+00	4.2E-01	8.4E-01	1.4E+00	6.3E-01
F15	4	Best	**9.6E-01**	9.7E-01	4.8E+00	1.4E+00	1.5E+00	1.3E+00	1.4E+00	9.7E-01	9.7E-01	1.0E+00	9.7E-01	9.7E-01
		Mean	2.4E+00	**9.9E-01**	5.1E+00	1.9E+00	1.9E+00	1.5E+00	1.5E+00	1.4E+00	1.5E+00	1.3E+00	1.3E+00	1.4E+00
		Std. dev	1.2E+00	**6.6E-02**	3.0E-01	7.7E-01	4.5E-01	1.5E-01	8.4E-02	2.7E-01	2.3E-01	1.9E-01	1.7E-01	2.0E-01
F16	2	Best	-1.0E+00	-1.0E+00	-1.0E+00	-1.0E+00	-1.0E+00	-1.0E+00	-1.0E+00	-1.0E+00	-1.0E+00	-1.0E+00	-1.0E+00	-1.0E+00
		Mean	**-1.0E+00**	-1.0E+00	-1.0E+00	**-1.0E+00**	-1.0E+00	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**
		Std. dev	7.1E-12	1.6E-05	1.9E-05	5.9E-07	4.1E-04	**5.3E-14**	2.0E-09	2.9E-09	5.9E-09	1.1E-09	3.0E-09	3.2E-09
F17	2	Best	4.0E-01	4.0E-01	4.0E-01	4.0E-01	**4.0E-01**	4.0E-01	4.0E-01	4.0E-01	4.0E-01	4.0E-01	4.0E-01	4.0E-01
		Mean	**4.0E-01**	4.0E-01	4.0E-01	**4.0E-01**	4.1E-01	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**
		Std. dev	6.8E-12	4.2E-06	1.3E-03	4.3E-10	1.3E-02	**2.3E-12**	1.7E-08	2.4E-09	1.1E-08	7.6E-09	2.8E-09	2.3E-09
F18	2	Best	3.0E+00	3.0E+00	3.0E+00	3.0E+00	3.0E+00	3.0E+00	3.0E+00	3.0E+00	3.0E+00	3.0E+00	3.0E+00	3.0E+00
		Mean	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**
		Std. dev	**0.0E+00**	**0.0E+00**	**0.0E+00**	2.6E-15	**0.0E+00**	**0.0E+00**	5.6E-15	7.5E-15	9.9E-15	8.4E-15	1.0E-14	1.3E-14
F19	3	Best	**-3.9E+00**	-6.8E-02	-3.9E+00	-6.8E-02	-6.8E-02	-6.8E-02	-1.9E+00	-6.8E-02	-1.9E+00	-1.9E+00	-6.8E-02	-1.9E+00
		Mean	-3.8E+00	-6.8E-02	**-3.9E+00**	-6.8E-02	-6.8E-02	-6.8E-02	-1.9E+00	-6.8E-02	-1.9E+00	-1.9E+00	-6.8E-02	-1.9E+00
		Std. dev	4.1E-02	**1.5E-17**	**3.5E-03**	**1.5E-17**	**1.5E-17**	**1.5E-17**	1.9E-12	**1.5E-17**	1.9E-12	2.8E-12	**1.5E-17**	3.1E-11
F20	6	Best	-3.3E+00	-5.1E-03	-3.3E+00	-5.1E-03	-5.1E-03	**-3.3E+00**	-1.2E+00	-5.1E-03	-1.2E+00	-1.2E+00	-5.1E-03	-1.2E+00
		Mean	-3.3E+00	-5.1E-03	**-3.3E+00**	-5.1E-03	-5.1E-03	-3.0E+00	-1.2E+00	-5.1E-03	-1.2E+00	-1.2E+00	-5.1E-03	-1.2E+00
		Std. dev	8.3E-02	**9.1E-19**	1.6E-02	**9.1E-19**	**9.1E-19**	6.5E-01	2.2E-12	**9.1E-19**	2.0E-12	2.9E-12	**9.1E-19**	6.4E-12
F21	4	Best	**-1.0E+01**	**-1.0E+01**	-5.1E+00	**-1.0E+01**	-3.9E+00	**-1.0E+01**	**-1.0E+01**	-5.1E+00	**-1.0E+01**	**-1.0E+01**	-5.1E+00	**-1.0E+01**
		Mean	-5.4E+00	-8.1E+00	-5.1E+00	-8.3E+00	-1.8E+00	-6.9E+00	**-1.0E+01**	-5.1E+00	**-1.0E+01**	**-1.0E+01**	-5.1E+00	**-1.0E+01**
		Std. dev	3.4E+00	2.6E+00	6.5E-07	2.9E+00	1.4E+00	3.6E+00	6.4E-06	**4.6E-07**	1.8E-05	7.4E-06	4.5E-06	5.6E-06
F22	4	Best	**-1.0E+01**	-1.0E+01	**-1.0E+01**	**-1.0E+01**	-4.8E+00	-1.0E+01	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**	**-1.0E+01**
		Mean	-7.9E+00	**-1.0E+01**	**-1.0E+01**	-9.0E+00	-2.2E+00	-9.0E+00	**-1.0E+01**	-9.1E+00	-9.9E+00	**-1.0E+01**	-8.4E+00	**-1.0E+01**
		Std. dev	3.3E+00	2.8E-05	2.7E-05	2.8E+00	9.7E-01	3.0E+00	5.1E-03	2.8E+00	1.7E+00	6.1E-06	3.2E+00	**5.1E-06**
F23	4	Best	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	-7.3E+00	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**	**-1.1E+01**
		Mean	-6.0E+00	-1.0E+01	**-1.1E+01**	-9.6E+00	-2.6E+00	-7.0E+00	**-1.1E+01**	**-1.1E+01**	-1.0E+01	**-1.1E+01**	**-1.1E+01**	-1.0E+01
		Std. dev	3.3E+00	1.7E+00	1.8E-05	2.0E+00	1.8E+00	3.8E+00	5.8E-06	**6.2E-08**	6.3E-01	4.6E-06	5.4E-05	1.7E+00
F24	10	Best	2.0E+00	**2.0E-07**	**2.0E-07**	**2.0E-07**	1.8E-02	**2.0E-07**	**2.0E-07**	**2.0E-07**	**2.0E-07**	**2.0E-07**	**2.0E-07**	**2.0E-07**
		Mean	6.0E+00	3.9E+00	1.7E+00	1.6E+00	3.4E+00	4.2E+00	**1.0E+00**	1.8E+00	1.2E+00	1.4E+00	2.5E+00	6.0E+00
		Std. dev	4.1E+00	3.4E+00	1.2E+00	1.3E+00	2.8E+00	3.1E+00	**2.2E-10**	1.1E+00	4.2E-01	8.4E-01	1.4E+00	6.3E-01
	30	Best	3.1E+03	1.2E+01	1.2E+05	2.8E+04	6.9E+03	3.3E+03	5.6E-06	**0.0E+00**	4.4E-03	7.8E-04	1.3E-03	1.4E-02
		Mean	9.9E+03	1.5E+02	1.2E+05	3.5E+04	1.8E+04	4.2E+03	7.3E-01	**2.9E-15**	4.4E+00	7.1E-01	1.6E+00	3.5E+00
		Std. dev	3.8E+03	3.0E+02	**0.0E+00**	4.0E+03	7.8E+03	7.6E+02	1.2E+00	2.0E-15	5.5E+00	1.6E+00	2.7E+00	5.9E+00
	60	Best	1.3E+05	3.6E+02	1.3E+06	6.2E+05	2.3E+05	7.0E+04	**4.2E-06**	5.3E-04	3.8E-03	7.4E-02	1.7E-02	1.5E-02
		Mean	1.8E+05	7.8E+03	1.3E+06	7.6E+05	3.8E+05	8.6E+04	1.0E+01	**1.2E+00**	1.4E+01	1.0E+01	1.2E+01	1.9E+01
		Std. dev	3.7E+04	1.3E+04	6.3E+03	6.7E+04	1.1E+05	1.1E+04	1.0E+01	**3.3E+00**	1.8E+01	1.3E+01	2.0E+01	2.1E+01
	90	Best	5.7E+05	7.3E+02	6.1E+06	3.5E+06	1.1E+06	2.9E+05	5.1E-03	**4.2E-05**	3.2E-01	3.3E-03	4.5E-01	1.4E-03
		Mean	7.6E+05	2.4E+04	6.1E+06	3.9E+06	2.0E+06	3.2E+05	**4.5E+00**	1.8E+02	2.6E+01	3.0E+01	4.9E+01	4.8E+01
		Std. dev	1.2E+05	6.9E+04	**0.0E+00**	2.1E+05	5.2E+05	2.6E+04	1.1E+01	5.7E+02	3.0E+01	2.8E+01	4.7E+01	7.6E+01
F25	10	Best	**9.6E-01**	9.7E-01	4.8E+00	1.4E+00	1.5E+00	1.3E+00	1.4E+00	9.7E-01	9.7E-01	1.0E+00	9.7E-01	9.7E-01
		Mean	2.4E+00	**9.9E-01**	5.1E+00	1.9E+00	1.9E+00	1.5E+00	1.5E+00	1.4E+00	1.5E+00	1.3E+00	1.3E+00	2.4E+00
		Std. dev	1.2E+00	**6.6E-02**	3.0E-01	7.7E-01	4.5E-01	1.5E-01	8.4E-02	2.7E-01	2.3E-01	1.9E-01	1.7E-01	2.0E-01
	30	Best	2.8E+03	1.2E+00	3.4E-01	1.6E+05	4.3E+04	7.7E+02	2.5E-01	7.7E-01	2.5E-01	**2.5E-01**	9.6E-01	2.5E-01
		Mean	1.8E+04	3.5E+02	6.2E+05	2.3E+05	3.1E+05	1.9E+03	3.9E-01	8.9E-01	**3.6E-01**	5.0E-01	1.1E+00	6.1E-01

TABLE 11. ***(Continued.)*** **The best, mean, and standard deviation of test function values found by the ALO, DA, MBO, MFO, SCA, SSA, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA on the test optimization functions with (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.**

		Std. dev	1.9E+04	8.6E+02	6.6E+05	6.8E+04	2.2E+05	1.0E+03	2.3E-01	**7.9E-02**	3.3E-01	3.4E-01	1.3E-01	4.3E-01
	60	Best	3.7E+04	4.7E+00	1.3E+06	4.0E+06	1.2E+06	4.2E+04	**2.5E-01**	9.2E-01	2.6E-01	2.6E-01	9.9E-01	2.6E-01
		Mean	1.3E+05	6.0E+02	1.3E+06	4.9E+06	2.2E+06	5.5E+04	**6.7E-01**	9.5E-01	8.8E-01	1.0E+00	1.2E+00	8.6E-01
		Std. dev	6.9E+04	1.4E+03	**2.5E-10**	7.4E+05	8.6E+05	7.2E+03	5.8E-01	2.2E-02	3.3E-01	4.9E-01	4.7E-01	4.0E-01
	90	Best	1.5E+05	1.5E+01	2.0E+03	1.0E+07	7.0E+06	1.2E+05	**2.5E-01**	9.7E-01	2.5E-01	2.6E-01	1.0E+00	2.5E-01
		Mean	3.6E+05	3.2E+04	2.0E+03	1.3E+07	9.8E+06	1.8E+05	8.8E-01	9.8E-01	1.2E+00	1.1E+00	1.6E+00	**8.2E-01**
		Std. dev	1.2E+05	8.7E+04	**2.4E-13**	1.2E+06	1.3E+06	3.1E+04	5.1E-01	7.4E-03	9.6E-01	4.2E-01	8.4E-01	9.0E-01
F26	10	Best	3.0E-05	3.0E-05	3.0E-05	3.0E-05	**1.0E-05**	3.0E-05	3.0E-05	3.0E-05	3.0E-05	3.0E-05	3.0E-05	3.0E-05
		Mean	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	-1.0E+00	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**	**-1.0E+00**
		Std. dev	7.1E-12	1.6E-05	1.9E-05	5.9E-07	4.1E-04	**5.3E-14**	2.0E-09	2.9E-09	5.9E-09	1.1E-09	3.0E-09	3.2E-09
	30	Best	9.2E+00	1.3E+00	1.7E+00	8.4E+01	3.1E+01	7.1E+00	3.3E-06	5.9E-01	**1.2E-07**	1.0E-06	6.6E-01	1.7E-06
		Mean	1.6E+01	6.1E+00	1.7E+00	9.9E+01	5.8E+01	1.5E+01	5.7E-03	1.0E+00	5.0E-03	**2.8E-03**	1.2E+00	6.9E-03
		Std. dev	7.2E+00	5.1E+00	**0.0E+00**	1.1E+01	1.6E+01	4.5E+00	8.1E-03	2.9E-01	5.4E-03	5.2E-03	2.5E-01	1.0E-02
	60	Best	2.8E+01	1.2E+01	2.8E-01	2.4E+02	1.4E+02	4.1E+01	9.8E-06	1.9E+00	5.7E-06	**8.9E-08**	2.9E+00	5.1E-06
		Mean	4.4E+01	2.7E+01	6.3E-01	3.2E+02	2.6E+02	5.1E+01	**2.9E-03**	2.6E+00	6.4E-03	3.5E-03	3.6E+00	6.3E-03
		Std. dev	1.1E+01	3.4E+01	7.8E-01	4.8E+01	6.3E+01	8.5E+00	**5.0E-03**	5.5E-01	6.0E-03	5.2E-03	4.7E-01	9.0E-03
	90	Best	6.0E+01	8.6E+00	4.2E-01	4.8E+02	3.7E+02	8.1E+01	1.8E-04	3.7E+00	**1.8E-08**	6.6E-06	5.6E+00	2.6E-05
		Mean	8.1E+01	2.0E+01	8.9E-01	5.3E+02	4.8E+02	9.6E+01	1.7E-02	4.3E+00	8.6E-03	7.7E-03	6.3E+00	**6.7E-03**
		Std. dev	2.4E+01	1.3E+01	1.2E+00	3.5E+01	6.2E+01	1.2E+01	2.0E-02	5.3E-01	9.1E-03	1.4E-02	4.3E-01	**7.9E-03**
F27	10	Best	**0.0E+00**	**0.0E+00**	6.0E-06	**0.0E+00**	4.1E-04	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**
		Mean	**4.0E-01**	4.0E-01	4.0E-01	**4.0E-01**	4.1E-01	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**	**4.0E-01**
		Std. dev	6.8E-12	4.2E-06	1.3E-03	4.3E-10	1.3E-02	**2.3E-12**	1.7E-08	2.4E-09	1.1E-08	7.6E-09	2.8E-09	2.3E-09
	30	Best	5.3E+01	1.2E+02	4.2E+01	**1.6E+01**	1.4E+02	9.0E+01	5.3E+01	5.6E+01	5.2E+01	5.1E+01	8.4E+01	5.3E+01
		Mean	**-2.5E+02**	-7.9E+01	-3.6E+01	-1.9E+02	-1.9E+01	-3.3E+01	-2.5E+02	-2.4E+02	-2.5E+02	-2.5E+02	-8.4E+01	-2.5E+02
		Std. dev	5.9E-01	1.5E+00	7.3E+01	3.1E+01	1.9E+01	3.8E+01	7.7E-01	4.2E+01	2.2E-01	**4.8E-03**	1.8E+01	7.5E-01
	60	Best	3.3E+02	**6.6E+01**	1.5E+02	1.2E+02	2.5E+02	2.7E+02	3.3E+02	3.3E+02	3.3E+02	3.3E+02	1.3E+02	3.3E+02
		Mean	**-5.3E+02**	-1.1E+02	-3.5E+02	1.0E+01	8.2E+01	9.1E+01	-5.3E+02	-4.0E+02	-5.3E+02	-5.3E+02	-4.8E+01	-5.3E+02
		Std. dev	3.9E-01	5.4E+01	4.8E+00	5.0E+01	2.5E+01	1.1E+01	6.9E-02	2.0E+02	**1.4E-03**	1.5E-02	1.4E+01	8.2E-02
	90	Best	6.1E+02	**1.1E+02**	4.8E+02	2.8E+02	3.4E+02	3.6E+02	6.1E+02	6.1E+02	6.1E+02	6.1E+02	1.1E+02	6.1E+02
		Mean	-8.1E+02	-6.1E+01	2.8E+02	3.3E+02	1.9E+02	1.8E+02	-8.1E+02	-5.4E+02	-8.1E+02	-8.1E+02	-5.9E+01	**-8.1E+02**
		Std. dev	5.0E-02	6.3E+01	**6.0E-14**	1.1E+02	3.3E+01	1.4E+01	2.1E-03	3.5E+02	1.1E-01	1.2E-02	2.2E+01	1.4E-01
F28	10	Best	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**
		Mean	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**	**3.0E+00**
		Std. dev	**0.0E+00**	**0.0E+00**	**0.0E+00**	2.6E-15	**0.0E+00**	**0.0E+00**	5.6E-15	7.5E-15	9.9E-15	8.4E-15	1.0E-14	1.3E-14
	30	Best	1.1E+02	8.2E-01	7.9E+03	1.1E+03	9.9E+02	2.7E+01	2.8E-06	**0.0E+00**	3.6E-05	2.2E-04	2.7E-05	3.1E-06
		Mean	1.5E+02	1.2E+01	8.7E+03	2.4E+03	1.6E+03	6.7E+01	4.3E-02	1.5E-01	1.4E-02	2.6E-02	**3.7E-03**	1.3E-02
		Std. dev	3.0E+01	2.9E+01	6.0E+02	1.1E+03	3.9E+02	2.8E+01	7.6E-02	4.9E-01	2.6E-02	4.4E-02	**7.7E-03**	2.1E-02
	60	Best	7.3E+02	1.7E+01	1.7E+04	1.3E+04	4.5E+03	3.0E+02	4.8E-07	**0.0E+00**	6.9E-06	4.1E-05	6.0E-07	3.1E-06
		Mean	1.4E+03	5.1E+02	1.7E+04	1.8E+04	1.3E+04	4.5E+02	4.0E-02	4.8E-02	6.5E-02	8.1E-02	**1.4E-02**	7.1E-02
		Std. dev	4.8E+02	1.3E+03	**3.8E-12**	2.7E+03	5.3E+03	7.6E+01	5.2E-02	1.5E-01	2.0E-01	1.0E-01	2.5E-02	1.5E-01
	90	Best	6.4E+02	3.2E+01	8.1E+03	2.9E+04	8.6E+03	5.2E+02	8.2E-06	**0.0E+00**	1.2E-05	7.8E-04	2.6E-04	7.8E-04
		Mean	2.1E+03	4.7E+02	8.7E+03	3.6E+04	1.7E+04	7.1E+02	1.6E-01	**2.6E-17**	5.0E-02	1.2E-01	5.3E-02	1.5E-01
		Std. dev	1.0E+03	8.3E+02	6.3E+02	3.4E+03	6.2E+03	1.0E+02	3.7E-01	**2.0E-17**	1.1E-01	1.5E-01	1.4E-01	2.4E-01
F29	10	Best	**0.0E+00**	3.8E+00	2.1E-03	3.8E+00	3.8E+00	3.8E+00	2.0E+00	3.8E+00	2.0E+00	2.0E+00	3.8E+00	2.0E+00
		Mean	-3.8E+00	**-6.8E-02**	**-3.9E+00**	**-6.8E-02**	**-6.8E-02**	**-6.8E-02**	**-1.9E+00**	**-6.8E-02**	**-1.9E+00**	**-1.9E+00**	**-6.8E-02**	-3.8E+00
		Std. dev	4.1E-02	**1.5E-17**	3.5E-03	**1.5E-17**	**1.5E-17**	**1.5E-17**	1.9E-12	**1.5E-17**	1.9E-12	2.8E-12	**1.5E-17**	3.1E-11
	30	Best	3.2E+00	7.2E+00	6.2E+00	**2.7E+00**	2.9E+00	6.4E+00	5.1E+00	3.3E+00	8.5E+00	7.1E+00	3.3E+00	6.4E+00
		Mean	-1.1E+01	**-1.6E+01**	-1.5E+01	-1.1E+01	-6.0E+00	-1.3E+01	-1.2E+01	-1.3E+01	-1.2E+01	-1.3E+01	-1.1E+01	-1.2E+01
		Std. dev	1.5E+00	1.1E+00	5.5E-01	1.3E+00	4.5E-01	1.7E+00	1.5E+00	**2.2E-01**	2.7E+00	1.8E+00	9.5E-01	1.6E+00
	60	Best	1.2E+01	1.4E+01	5.1E+00	1.1E+01	**1.6E+00**	1.4E+01	1.8E+01	1.5E+01	1.7E+01	1.4E+01	1.1E+01	1.9E+01
		Mean	-1.8E+01	-2.2E+01	-1.4E+01	-1.6E+01	-1.0E+01	-2.1E+01	-2.1E+01	**-2.4E+01**	-2.2E+01	-2.0E+01	-1.8E+01	-2.1E+01
		Std. dev	1.8E+00	6.5E-01	**5.4E-01**	2.5E+00	5.6E-01	1.7E+00	4.6E+00	7.3E-01	4.0E+00	2.6E+00	1.6E+00	4.5E+00
	90	Best	1.7E+01	2.3E+01	2.6E+01	1.3E+01	**5.2E+00**	2.3E+01	3.0E+01	2.4E+01	2.7E+01	2.9E+01	1.7E+01	2.7E+01
		Mean	-2.3E+01	-3.2E+01	**-3.5E+01**	-2.0E+01	-1.4E+01	-3.0E+01	-3.4E+01	-2.7E+01	-3.3E+01	-3.3E+01	-2.3E+01	-3.3E+01
		Std. dev	1.7E+00	1.7E+00	2.0E+00	1.9E+00	**7.0E-01**	2.2E+00	3.1E+00	4.6E+00	2.6E+00	3.9E+00	2.6E+00	2.6E+00
F30	10	Best	4.0E-04	3.3E+00	6.4E-04	3.3E+00	3.3E+00	**3.8E-04**	2.2E+00	3.3E+00	2.2E+00	2.2E+00	3.3E+00	2.2E+00
		Mean	-3.3E+00	**-5.1E-03**	**-3.3E+00**	**-5.1E-03**	**-5.1E-03**	-3.0E+00	**-1.2E+00**	**-5.1E-03**	**-1.2E+00**	**-1.2E+00**	**-5.1E-03**	-3.3E+00
		Std. dev	8.3E-02	**9.1E-19**	1.6E-02	**9.1E-19**	**9.1E-19**	6.5E-01	2.2E-12	**9.1E-19**	2.0E-12	2.9E-12	**9.1E-19**	6.4E-12
	30	Best	4.1E+02	3.6E+02	7.0E+01	4.2E+02	1.5E+02	2.7E+02	3.4E-10	**0.0E+00**	8.5E-04	1.5E-02	3.1E-02	2.6E-05
		Mean	5.1E+02	3.6E+02	7.4E+01	4.7E+02	2.4E+02	4.3E+02	7.0E-02	**2.3E-04**	8.6E-01	4.7E-01	1.4E+00	5.7E+01
		Std. dev	7.4E+01	6.1E+00	1.3E+01	4.3E+01	5.9E+01	8.4E+01	1.5E-01	**7.4E-04**	1.2E+00	7.1E-01	1.3E+00	5.5E+01
	60	Best	4.9E+02	1.0E+03	2.0E+03	1.4E+03	5.4E+02	1.1E+03	6.2E-04	**0.0E+00**	8.0E-01	4.8E-03	4.7E-02	6.1E-01
		Mean	8.9E+02	1.0E+03	2.0E+03	1.5E+03	8.0E+02	1.4E+03	8.2E+01	**5.9E+00**	1.6E+02	3.6E+02	2.4E+02	6.1E+02
		Std. dev	2.0E+02	**6.3E+00**	9.4E+01	6.7E+01	3.0E+02	2.1E+02	2.4E+02	9.2E+00	3.1E+02	4.6E+02	3.4E+02	4.5E+02
	90	Best	1.1E+03	1.3E+03	3.5E+02	1.9E+03	1.0E+03	2.0E+03	8.9E-04	**0.0E+00**	3.2E+00	9.0E+00	2.0E-02	2.6E+00
		Mean	1.4E+03	1.3E+03	3.9E+05	2.4E+03	1.5E+03	2.5E+03	2.6E+02	**1.3E-17**	5.8E+02	5.3E+02	1.4E+02	8.4E+02
		Std. dev	2.2E+02	1.9E+01	9.8E+05	2.7E+02	3.2E+02	3.3E+02	5.4E+02	**7.3E-18**	7.2E+02	8.1E+02	2.1E+02	8.8E+02
F31	10	Best	**0.0E+00**	**0.0E+00**	5.1E+00	1.1E-03	6.3E+00	**0.0E+00**	**0.0E+00**	5.1E+00	**0.0E+00**	**0.0E+00**	5.1E+00	**0.0E+00**

TABLE 11. ***(Continued.)*** **The best, mean, and standard deviation of test function values found by the ALO, DA, MBO, MFO, SCA, SSA, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA on the test optimization functions with (2, 3, 4, 6, 10, 30, 60 and 90) dimensions.**

		Mean	-5.4E+00	-8.1E+00	**-5.1E+00**	-8.3E+00	-1.8E+00	-6.9E+00	**-1.0E+01**	**-5.1E+00**	**-1.0E+01**	**-1.0E+01**	-5.1E+00	-5.4E+00
		Std. dev	3.4E+00	2.6E+00	6.5E-07	2.9E+00	1.4E+00	3.6E+00	6.4E-06	**4.6E-07**	1.8E-05	7.4E-06	4.5E-06	5.6E-06
	30	Best	4.2E+09	**0.0E+00**	5.4E+08	1.3E+10	2.2E+09	6.1E+08	5.4E+01	**0.0E+00**	1.8E-03	1.4E+03	5.5E+01	8.0E+02
		Mean	6.0E+09	1.5E+08	5.5E+08	2.3E+10	7.1E+09	1.2E+09	3.8E+05	5.1E+05	9.3E+05	3.3E+05	**1.3E+05**	8.7E+05
		Std. dev	1.8E+09	4.8E+08	1.6E+07	5.9E+09	3.0E+09	2.6E+08	4.3E+05	1.5E+06	1.4E+06	4.3E+05	**2.4E+05**	1.4E+06
	60	Best	1.3E+10	4.2E+06	1.3E+11	6.7E+10	2.1E+10	4.7E+09	1.6E+01	**1.5E+01**	9.1E+02	7.7E+02	6.7E+03	1.4E+04
		Mean	1.9E+10	3.9E+08	1.3E+11	8.8E+10	4.3E+10	5.2E+09	6.0E+05	**1.9E+04**	6.3E+05	1.3E+06	4.5E+05	6.3E+05
		Std. dev	3.5E+09	1.0E+09	**0.0E+00**	9.4E+09	1.7E+10	3.8E+08	8.5E+05	2.1E+04	4.9E+05	1.7E+06	6.3E+05	6.8E+05
	90	Best	2.2E+10	1.9E+07	2.2E+11	1.8E+11	5.9E+10	1.2E+10	1.4E+00	**0.0E+00**	6.0E+02	1.4E+04	3.8E+01	2.0E+05
		Mean	2.9E+10	8.3E+08	2.2E+11	1.9E+11	9.2E+10	1.2E+10	3.0E+05	**7.2E-10**	1.0E+06	7.9E+05	4.8E+05	1.1E+06
		Std. dev	4.3E+09	2.0E+09	**0.0E+00**	7.5E+09	2.3E+10	7.5E+08	3.7E+05	3.5E-10	9.2E+05	8.2E+05	4.6E+05	9.7E+05
F32	10	Best	**0.0E+00**	**0.0E+00**	**0.0E+00**	6.0E-04	5.6E+00	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**
		Mean	-7.9E+00	**-1.0E+01**	**-1.0E+01**	-9.0E+00	-2.2E+00	-9.0E+00	-1.0E+01	-9.1E+00	-9.9E+00	**-1.0E+01**	-8.4E+00	-7.9E+00
		Std. dev	3.3E+00	2.8E-05	2.7E-05	2.8E+00	9.7E-01	3.0E+00	5.1E-03	2.8E+00	1.7E+00	6.1E-06	3.2E+00	**5.1E-06**
	30	Best	6.9E+02	9.9E+02	1.1E+03	7.6E+02	**5.4E+02**	8.4E+02	1.1E+03	9.4E+02	1.1E+03	1.1E+03	9.3E+02	1.1E+03
		Mean	-7.1E+02	-9.8E+02	-9.9E+02	-7.5E+02	-5.0E+02	-8.3E+02	**-1.2E+03**	-9.7E+02	-1.2E+03	**-1.2E+03**	-9.1E+02	**-1.2E+03**
		Std. dev	1.1E+01	7.7E+01	1.8E+02	4.6E+01	4.2E+01	4.0E+01	5.5E-02	8.9E+00	7.5E-02	5.7E-02	2.9E+01	**2.8E-02**
	60	Best	1.2E+03	1.9E+03	2.3E+03	1.1E+03	**9.8E+02**	1.5E+03	2.3E+03	1.9E+03	2.3E+03	2.3E+03	1.6E+03	2.3E+03
		Mean	-1.2E+03	-1.9E+03	-2.4E+03	-1.0E+03	-8.9E+02	-1.5E+03	-2.4E+03	-1.9E+03	**-2.4E+03**	-2.4E+03	-1.6E+03	-2.4E+03
		Std. dev	3.3E+01	8.3E+01	2.8E-01	8.1E+01	7.0E+01	6.4E+01	1.1E-01	1.2E+02	**3.3E-02**	1.0E-01	4.6E+01	2.1E-01
	90	Best	1.7E+03	2.9E+03	2.4E+03	**1.3E+03**	1.6E+03	2.0E+03	3.5E+03	2.5E+03	3.5E+03	3.5E+03	2.1E+03	3.5E+03
		Mean	-1.6E+03	-2.9E+03	-2.5E+03	-1.3E+03	-1.1E+03	-2.0E+03	-3.5E+03	-2.5E+03	**-3.5E+03**	-3.5E+03	-2.1E+03	-3.5E+03
		Std. dev	4.9E+01	2.0E+02	**0.0E+00**	7.6E+01	1.9E+02	7.0E+01	1.1E-01	1.7E+02	4.9E-02	7.2E-02	4.9E+01	6.1E-02
F33	10	Best	**0.0E+00**	**0.0E+00**	**0.0E+00**	1.4E-03	3.2E+00	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**	**0.0E+00**
		Mean	-6.0E+00	-1.0E+01	**-1.1E+01**	-9.6E+00	-2.6E+00	-7.0E+00	**-1.1E+01**	**-1.1E+01**	-1.0E+01	**-1.1E+01**	**-1.1E+01**	-6.0E+00
		Std. dev	3.3E+00	1.7E+00	1.8E-05	2.0E+00	1.8E+00	3.8E+00	5.8E-06	**6.2E-08**	6.3E-01	4.6E-06	5.4E-05	1.7E+00
	30	Best	2.4E-05	9.8E-08	1.2E-03	2.7E-02	6.8E-10	1.3E-04	5.8E-14	**0.0E+00**	4.1E-13	1.5E-11	2.0E-11	3.3E-12
		Mean	5.8E-04	3.6E-05	4.5E+00	1.8E-01	1.6E-03	2.8E-03	**3.8E-08**	5.4E-08	7.7E-08	7.3E-08	1.7E-07	3.9E-07
		Std. dev	9.3E-04	7.5E-05	1.6E+00	1.2E-01	2.2E-03	3.0E-03	**9.7E-08**	1.7E-07	1.8E-07	1.6E-07	2.6E-07	1.2E-06
	60	Best	5.1E-05	1.5E-08	3.5E-09	2.2E+00	1.0E-06	2.1E-03	3.6E-12	**0.0E+00**	2.0E-11	2.2E-12	9.0E-13	2.0E-12
		Mean	1.9E-03	2.8E-04	7.2E-07	3.8E+00	1.5E-02	1.5E-02	8.4E-09	7.8E-09	2.0E-07	1.8E-08	**6.1E-09**	5.4E-08
		Std. dev	2.2E-03	8.1E-04	1.3E-06	1.1E+00	1.7E-02	7.8E-03	1.7E-08	1.7E-08	5.5E-07	2.5E-08	**1.2E-08**	7.1E-08
	90	Best	3.5E-04	3.6E-09	**1.1E-27**	6.4E+00	6.6E-05	1.9E-02	1.6E-13	2.0E-14	3.5E-14	3.3E-11	7.6E-16	3.6E-13
		Mean	5.0E-03	4.8E-04	**1.1E-11**	8.6E+00	1.3E-02	5.3E-02	7.2E-09	5.6E-08	4.9E-08	1.3E-07	7.6E-10	2.8E-09
		Std. dev	6.3E-03	1.4E-03	**3.4E-11**	1.0E+00	2.0E-02	2.8E-02	1.7E-08	7.7E-08	6.7E-08	3.0E-07	2.3E-09	4.6E-09
F34	10	Best	1.4E+01	1.0E+02	1.1E-05	2.7E+01	3.4E-01	2.9E+00	1.0E-08	**0.0E+00**	5.0E-06	2.4E-04	2.3E-04	4.6E-06
		Mean	9.6E+01	1.0E+02	4.0E+02	8.2E+01	1.2E+01	1.5E+01	1.3E-02	8.3E-03	6.2E-02	5.6E-03	1.5E-03	5.8E-02
		Std. dev	6.3E+01	2.2E+00	8.0E+02	3.8E+01	1.1E+01	1.2E+01	3.3E-02	1.9E-02	7.1E-02	1.0E-02	**1.5E-03**	1.2E-01
	30	Best	2.1E+04	2.8E+01	3.5E+05	9.5E+04	1.7E+04	4.9E+03	6.2E-05	**3.5E-05**	3.4E-03	2.3E-02	7.3E-03	5.3E-03
		Mean	3.6E+04	5.7E+02	3.5E+05	1.3E+05	5.5E+04	8.4E+03	1.4E+00	**4.6E-01**	2.9E+00	3.9E+00	1.5E+00	1.4E+00
		Std. dev	9.1E+03	8.8E+02	4.2E+03	1.6E+04	2.7E+04	2.4E+03	1.9E+00	**8.9E-01**	5.8E+00	5.0E+00	2.1E+00	1.5E+00
	60	Best	2.1E+05	6.5E+01	2.2E+00	1.0E+06	3.9E+05	6.7E+04	6.0E-04	**0.0E+00**	2.6E-01	6.2E-02	1.3E-01	3.2E-02
		Mean	2.5E+05	3.5E+04	1.1E+02	1.2E+06	5.8E+05	7.6E+04	**4.2E+00**	6.2E+00	1.4E+01	8.6E+00	9.2E+00	1.1E+01
		Std. dev	5.0E+04	6.5E+04	1.1E+02	9.5E+04	1.4E+05	7.6E+03	9.2E+00	8.0E+00	1.3E+01	1.3E+01	1.0E+01	8.3E+00
	90	Best	4.4E+05	1.5E+01	3.8E+03	3.0E+06	3.1E+05	1.6E+05	2.3E-04	**0.0E+00**	1.9E-02	8.6E-01	2.3E-03	5.4E-01
		Mean	6.2E+05	1.9E+04	6.5E+03	3.3E+06	1.6E+06	1.9E+05	**7.9E+00**	2.6E+01	1.4E+01	1.7E+01	1.8E+01	2.6E+01
		Std. dev	1.7E+05	5.4E+04	4.2E+03	1.9E+05	6.0E+05	1.3E+04	1.3E+01	7.3E+01	1.3E+01	2.1E+01	2.6E+01	2.5E+01

Moreover, Table 11 presents the results on F1- F13 and F24 - F34 in 10, 30, 60, and 90 of the high-dimensional optimization functions. It was the fastest among the algorithms to get the global optimum on functions: F1, F2, F3, F4, F5, F6, F7, F9, F10, F11, F12, F13, F24, F25, F26, F28, F30, F31, F33, and F34, with the exception of some of the functions, that the proposed algorithm has not been able to win in all statistical measurements (best, mean, and standard deviation), particularly functions F8, F25, F26, F27, F29, and F32.

Figure 8(a) show the results for F11 Griewank function. From Figure 8(a), it is obvious that E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA outperform all other methods during the whole progress of optimization in these test optimization functions. Although slower, DA and MBO performs the second best, after the six proposed algorithms, at finding the global minimum. In many cases, few algorithms such as DA, MBO, MFO, SCA, and SSA seem to trap into local optima and fail to find the best solution. The convergence trends of test optimization functions with 30 dimensions are depicted in Figures 8(b-c). It can be disclosed from these figures that the six proposed algorithms have fast convergence compared with other optimization algorithms. Figure 8(c) shows the optimization process against

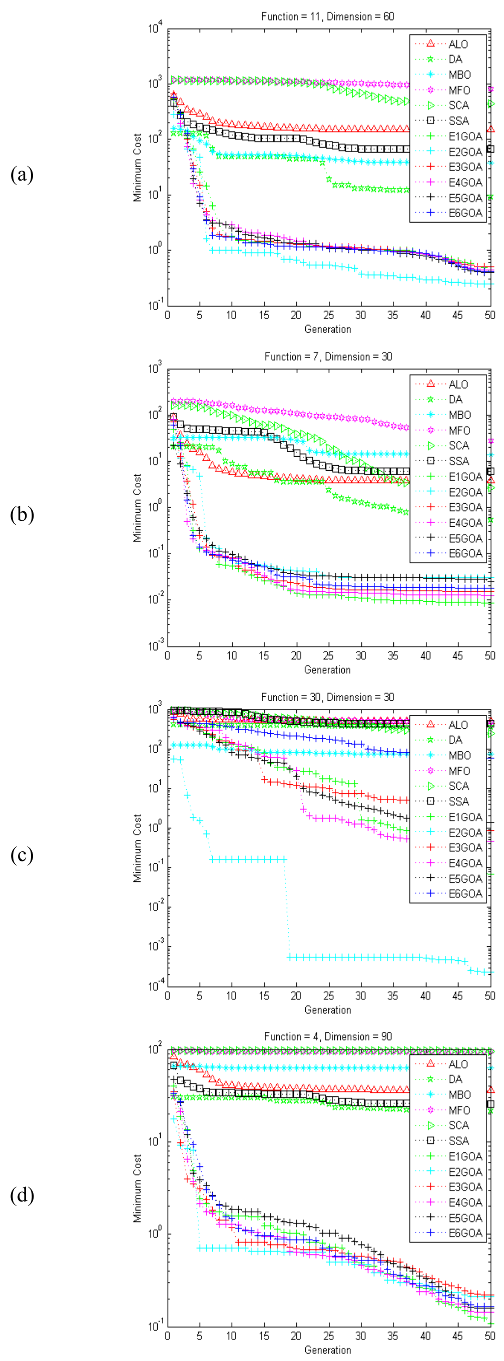

FIGURE 8. Convergence curves of GOA, E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA against 4 functions (A-F11, B- F7, C-F30, D-F4) with 30, and 90 dimensions over 50 generations.

(F30: Dim = 30), the E2GOA converges faster than E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA. ALO, DA, MBO, SCA, and SSA algorithms fall into the trap of the local optimum and fail to find the best solution. Figure 8(d) display the result for F4 Schwefel 2.21 function with 90 dimension. In terms of convergence, the six proposed algorithms converge very fast in the case of all functions compared to the other algorithms. In contrast, ALO, E2GOA, and E5GOA reached the global optimum in the 3, 5, and 10 iterations, respectively. The DA, MBO, MFO, and SSA seem to trap into local optima and fail to find the best solution. From Figure 8 (in most functions) it is noticed that E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA have a close performance in terms of both end result and convergence path.

1) COMPARISON THE EGOA WITH THE OTHER IMPROVED METHODS BASED ON GOA

This section presents the comparison results between EGOA and the other six enhancements to the original GOA algorithm, which have been analyzed in the related work section. The setup of this set of comparisons is the same as the previous experiments, and the purpose of these as well as later experiments is to assess EGOA in comparison with a wide variety of algorithms and to reach a more accurate evaluation. We start the section by briefly introducing all the algorithms used in our comparison. For more details, we refer the reader to the original references of the algorithms. We notice from the discussion in the related work section that the different works try to improve the original algorithm by adding certain operators in order to establish a better balance between exploration and exploitation. This goal is evident in the proposal literature. Hence, our proposal is similarly pursuing the same idea sought by other researchers, but by using a new and more appropriate mutation operator this time around. It is this unique adjustment that has empowered EGOA over the other algorithms as shown by the results presented later in the section. For the purpose of the evaluation process, we list all the control variables used by the algorithms in Table 12. We report the results of over 30 implementation runs for each algorithm on each benchmark function listed in Table 4.

TABLE 12. The parameter settings.

Algorithm	Parameter
IGOA [45]	$C_{max} = 1$, C_{min}=0.00004, β=1.5
OBLGOA [44]	$C_{max} = 1$, C_{min} =0.00004
CGOA [46]	$C_{max} = 1$, C_{min} =0.00004
IGOA [48]	$C_{max} = 1$, C_{min} =0.00004, β=1.5
IGOA [49]	$C_{max} = 1$, C_{min} =0.00004
CGOA [47]	$C_{max} = 1.5$, C_{min} =0.0009
Our proposal	$C_{max} = 1$, C_{min} =0.00004

The comparison results of our algorithm, as well as other algorithms, are shown in Table 13. The variables of the optimization functions were standardized across all algorithms. The results of the other algorithms were directly taken from the original reports which contain all necessary details which can be consulted by the reader. Table 13 shows the performance of the six proposed algorithms (E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA) compared with the optimization methods mentioned in section II. Table 13 demonstrates the best possible solution (Best) and the average result (Mean) obtained by each method. So, each benchmark function has two rows for each outcome. The values in the

TABLE 13. Comparisons of the related works and the six proposed algorithms (EGOA) and GOA on benchmark function.

Function	Statistical parameters	IGOA [45]	OBLGOA [44]	CGOA [46]	IGOA [48]	IGOA [49]	CGOA [47]	GOA	E1GOA	E2GOA	E3GOA	E4GOA	E5GOA	E6GOA
sphere	Best	5.8E-08	1.3E-08	-	8.0E-16	-	-	1.2E+02	1.4E-08	**0.0E+00**	1.9E-05	1.6E-01	2.3E-06	2.2E-02
	Mean	1.9E-10	3.0E-06	2.1E+01	3.4E-15	-	**0.0E+00**	7.1E-02	5.9E-02	**4.3E-16**	9.1E-02	6.5E-02	5.4E-02	4.9E-02
schwefel 2.22	Best	-	2.5E-05	-	9.0E-09	2.5E-05	-	5.7E-01	2.4E-04	**0.0E+00**	3.4E-03	6.4E-03	2.3E-03	8.7E-03
	Mean	-	9.4E-05	3.9E+01	2.4E-08	2.6E-05	-	5.7E-01	5.8E-02	**9.1E-05**	3.2E-02	5.7E-02	5.2E-02	1.2E-01
schwefel 1.2	Best	-	3.1E-07	-	8.7E-12	-	-	9.1E+01	2.2E-07	**0.0E+00**	3.1E-02	9.7E-02	1.8E-02	2.6E-03
	Mean	-	3.0E-04	2.5E+02	1.2E-02	-	-	9.1E+01	1.2E-01	**2.1E-04**	5.5E-01	1.2E+00	7.2E-01	2.5E+00
schwefel 2.21	Best	-	4.3E-05	-	9.7E-07	-	-	4.0E-01	1.2E-02	**0.0E+00**	1.3E-02	4.9E-03	7.7E-03	6.0E-03
	Mean	-	**3.4E-04**	1.6E+01	9.7E-07	-	-	4.0E-01	1.3E-01	2.1E-02	1.3E-01	7.6E-02	1.2E-01	1.0E-01
Rosenbrock	Best	1.9E+03	8.5E+00	-	2.6E+01	**0.0E+00**	-	4.5E+02	3.9E-02	8.3E+00	7.9E-03	1.9E-03	8.9E+00	1.4E-03
	Mean	8.3E+00	8.8E+00	2.7E+02	2.6E+01	2.0E-07	9.5E+00	4.5E+02	3.9E+00	9.0E+00	4.2E+00	4.3E+00	9.5E+00	4.8E+00
Step	Best	-	-	-	**5.3E-07**	-	-	3.5E-02	7.5E-04	3.1E-03	6.7E-05	3.5E-05	3.2E-02	6.8E-04
	Mean	-	-	1.8E+01	1.4E-06	-	-	3.5E-02	1.0E-01	3.2E-02	7.6E-02	8.1E-02	1.4E-01	1.2E-01
Quantic	Best	-	5.4E-05	-	**1.1E-05**	-	-	1.5E-02	1.2E-03	4.8E-04	4.3E-04	5.6E-04	4.9E-04	7.1E-04
	Mean	-	**2.2E-04**	1.2E+00	1.0E-03	-	-	4.0E+02	6.7E-03	1.5E-02	1.3E-02	7.0E-03	1.7E-02	1.7E-02
schwefel 2.26	Best	-	-3.6E+03		-9.0E+03	-		-2.5E+03	**-4.2E+02**	**-3.1E+03**	**-4.2E+02**	**-4.2E+02**	**-3.3E+03**	**-4.2E+02**
	Mean	-	-3.0E+03	**-7.3E+03**	-7.6E+03	-		-2.5E+03	**-4.2E+02**	-3.1E+03	**-4.2E+02**	**-4.2E+02**	**-3.1E+03**	**-4.2E+02**
Rastrigin	Best	-	5.7E-09	-	**0.0E+00**	-	-	3.0E+01	1.1E-09	**0.0E+00**	2.6E-05	1.2E-05	9.3E-05	1.9E-07
	Mean	-	2.6E-08	1.2E+02	**0.0E+00**	-	**0.0E+00**	3.0E+01	3.9E-04	**0.0E+00**	1.0E+00	3.6E-01	2.2E-03	1.2E+00
Ackley	Best	-	4.5E-05	-	8.6E-09	8.6E-08	-	4.2E-01	9.8E-04	**8.9E-16**	5.7E-04	2.9E-03	1.7E-04	3.7E-03
	Mean	-	2.0E-04	1.2E+01	1.3E-08	3.4E-07	**0.0E+00**	4.2E-01	1.9E-01	**9.8E-09**	2.7E-01	2.8E-01	9.0E-02	1.5E-01
Griewank	Best	1.9E-08	9.1E-06	-	6.7E-16	2.5E-02	-	2.8E-01	4.5E-05	**0.0E+00**	6.2E-03	7.8E-03	3.6E-04	8.4E-06
	Mean	**5.7E-10**	9.3E-05	1.3E+00	5.5E-15	1.2E-01	-	2.8E-01	2.7E-01	1.4E-01	1.8E-01	2.7E-01	2.0E-01	1.9E-01
Penalty 1	Best	-	1.8E-03	-	**4.6E-08**	-	-	1.9E-02	7.1E-04	8.2E-05	1.5E-05	5.7E-04	8.7E-03	1.2E-04
	Mean	-	2.1E-02	1.6E+01	**8.9E-08**	-	-	1.9E-02	9.8E-03	1.6E-03	4.5E-03	1.1E-02	8.0E-02	5.9E-03
Penalty 2	Best	-	1.6E-02	-	**4.9E-07**	-	-	2.1E-02	1.9E-03	1.3E-04	1.4E-04	2.3E-05	1.4E-02	8.1E-04
	Mean	-	6.9E-02	6.7E+01	1.4E-02	-	-	2.1E-02	2.5E-02	**2.4E-03**	1.8E-02	2.1E-02	3.7E-02	1.7E-02
Sum squares	Best	-	-	-	-	6.6E-11	-	5.7E+02	5.6E-06	**0.0E+00**	4.4E-03	7.8E-04	1.3E-03	1.4E-02
	Mean	-	-	-	-	8.4E-06	-	5.7E+02	7.3E-01	2.9E-15	4.4E+00	7.1E-01	1.6E+00	3.5E+00
Dixon-price	Best	-	-	-	-	-	-	3.9E+01	2.5E-01	7.7E-01	2.5E-01	2.5E-01	9.6E-01	2.5E-01
	Mean	-	-	-	-	-	-	1.0E+00	1.5E+00	1.4E+00	1.5E+00	1.3E+00	1.3E+00	1.4E+00
Levy	Best	-	-	-	-	-	-	6.3E+01	3.3E-06	5.9E-01	1.2E-07	1.0E-06	6.6E-01	1.7E-06
	Mean	-	-	-	-	-	-	3.0E+00	3.0E-05	3.0E-05	3.0E-05	3.0E-05	3.0E-05	3.0E-05
Trid	Best	-	-	-	-	-	-	1.8E+02	5.3E+01	5.6E+01	5.2E+01	5.1E+01	8.4E+01	5.3E+01
	Mean	-	-	-	-	-	-	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Powell	Best	-	-	-	-	-	-	4.5E+01	2.8E-06	0.0E+00	3.6E-05	2.2E-04	2.7E-05	3.1E-06
	Mean	-	-	-	-	-	-	-1.0E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Michalewicz	Best	-	-	-	-	-	-	**-1.1E+01**	5.1E+00	3.3E+00	8.5E+00	7.1E+00	3.3E+00	6.4E+00
	Mean	-	-	-	-	-	-	3.8E+00	2.0E+00	3.8E+00	2.0E+00	2.0E+00	3.8E+00	2.0E+00
Zakharov	Best	-	-	-	-	3.7E-13	-	3.6E+02	3.4E-10	**0.0E+00**	8.5E-04	1.5E-02	3.1E-02	2.6E-05
	Mean	-	-	-	-	1.7E-07	-	1.0E+00	2.0E-03	7.5E-18	8.8E-02	2.1E-02	5.4E-02	1.4E-01
Bent cigar	Best	-	-	-	-	-	-	1.5E+08	5.4E+01	0.0E+00	1.8E-03	1.4E+03	5.5E+01	8.0E+02
	Mean	-	-	-	-	-	-	5.1E-00	0.0E+00	5.1E+00	0.0E+00	0.0E+00	5.1E+00	0.0E+00
Styblinski -Tang	Best	-	-	-	-	-	-	-9.2E+02	1.1E+03	9.4E+02	1.1E+03	1.1E+03	9.3E+02	1.1E+03
	Mean	-	-	-	-	-	-	0.0E+00	1.6E-03	1.3E+00	5.3E-01	0.0E+00	2.0E+00	0.0E+00
Sum of differe-nt powers	Best	-	-	-	-	-	-	1.3E+01	5.8E-14	0.0E+00	4.1E-13	1.5E-11	2.0E-11	3.3E-12
	Mean	-	-	-	-	-	-	1.4E+01	0.0E+00	0.0E+00	2.0E-01	0.0E+00	0.0E+00	5.4E-01
Rotated Hyper-ellipsoid	Best	-	-	-	-	-	-	3.6E+02	6.2E-05	3.5E-05	3.4E-03	2.3E-02	7.3E-03	5.3E-03
	Mean	-	-	-	-	-	-	3.6E+02	1.3E-02	8.3E-03	6.2E-02	5.6E-03	1.5E-03	5.8E-02

last two rows of this table show the total number of benchmark functions for which the specific optimization algorithm performs better than other methods; the values are marked in bold. If a particular method does not provide a result for the benchmark function, it is signified by a hyphen "–". It is evident from the obtained results that all EGOA algorithms have significantly better performed over their rivals across all functions. Especially E2GOA which has exceptionally achieved the best performance overall rest in terms of finding the global minimum and average values for most of the functions.

2) COMPUTATION TIME

In this section, we present the computation time of the proposed algorithms against the original algorithm (GOA), which is considered another important factor. To further clarify the advantage of the proposed algorithm over the original algorithm we calculated the computation time between the two. Due to the limited space in the article, we would limit the comparisons to 30 dimensions. Table 14 demonstrates the computation time for the EGOAs against the original GOA. The results are shown in Table 14. The new algorithms were clearly the fastest optimization methods, whereas the original algorithm (GOA) was the slowest. Another conclusion of the above experiment was that the fitness function evaluation of real-world applications is the most expensive part of the optimization algorithm. Therefore, the proposed algorithm (EGOAs) aims to make a trade-off between computational requirements and finding near-optimal solutions.

D. SCENARIO 4: PERFORMANCE OF EGOAMLP AGAINST THE SpamBase DATASET

The experimental test of scenario 4 verified the effectiveness of the proposed EGOAMLP models and other metaheuristic

TABLE 14. Computation time for thirteen benchmark functions in 30 dimensions.

Function	GOA	E1GOA	E2GOA	E3GOA	E4GOA	E5GOA	E6GOA
1	1.1E+02	1.0E+02	1.0E+02	1.0E+02	1.0E+02	1.0E+02	1.0E+02
2	1.0E+02	9.9E+01	9.9E+01	9.8E+01	9.8E+01	9.9E+01	1.0E+02
3	1.1E+02	1.0E+02	1.0E+02	1.0E+02	1.0E+02	1.0E+02	1.0E+02
4	1.1E+02	1.0E+02	9.8E+01	9.9E+01	9.9E+01	9.9E+01	9.7E+01
5	1.1E+02	9.9E+01	9.8E+01	9.8E+01	9.7E+01	9.8E+01	1.0E+02
6	1.0E+02	9.8E+01	9.9E+01	9.6E+01	9.6E+01	9.9E+01	9.8E+01
7	1.0E+02	9.9E+01	9.9E+01	9.8E+01	1.0E+02	1.0E+02	1.0E+02
8	1.0E+02	9.8E+01	9.8E+01	9.9E+01	9.9E+01	1.0E+02	1.0E+02
9	1.0E+02	9.8E+01	1.0E+02	9.8E+01	9.9E+01	1.0E+02	9.8E+01
10	1.0E+02	1.0E+02	9.8E+01	9.9E+01	1.0E+02	1.0E+02	9.8E+01
11	1.1E+02	1.2E+02	1.5E+02	1.4E+02	1.2E+02	1.3E+02	1.0E+02
12	1.5E+02	1.7E+02	1.4E+02	1.5E+02	1.6E+02	2.1E+02	2.3E+02
13	1.1E+02	1.1E+02	1.0E+02	9.2E+01	9.4E+01	9.4E+01	9.6E+01
14	1.0E+02	1.3E+02	1.4E+02	1.4E+02	1.5E+02	1.5E+02	1.5E+02
15	7.5E+01	7.2E+01	7.2E+01	7.3E+01	7.2E+01	7.1E+01	7.3E+01
16	1.7E+02	1.3E+02	1.3E+02	1.1E+02	1.1E+02	1.1E+02	1.0E+02
17	1.1E+02	9.8E+01	9.7E+01	9.9E+01	9.6E+01	9.3E+01	1.1E+02
18	1.6E+02	1.4E+02	1.6E+02	1.3E+02	1.2E+02	1.1E+02	1.1E+02
19	6.5E+01	5.9E+01	6.1E+01	6.9E+01	6.3E+01	6.5E+01	6.4E+01
20	1.1E+02	1.1E+02	1.1E+02	1.0E+02	9.5E+01	9.6E+01	1.3E+02
21	7.7E+01	7.1E+01	7.2E+01	7.1E+01	7.4E+01	7.1E+01	7.4E+01
22	8.4E+01	9.8E+01	8.4E+01	8.0E+01	1.1E+02	1.5E+02	1.6E+02
23	8.1E+01	9.0E+01	7.8E+01	8.7E+01	9.0E+01	8.6E+01	8.4E+01
24	1.1E+02	1.1E+02	1.2E+02	1.2E+02	1.2E+02	1.2E+02	1.1E+02
25	1.3E+02	1.2E+02	8.0E+01	7.8E+01	8.2E+01	7.8E+01	1.0E+02
26	1.1E+02	1.0E+02	9.9E+01	9.9E+01	1.3E+02	1.4E+02	8.3E+01
27	6.9E+01	7.3E+01	7.2E+01	7.3E+01	7.1E+01	7.1E+01	7.2E+01
28	1.1E+02	9.4E+01	1.0E+02	1.6E+02	1.4E+02	9.8E+01	9.4E+01
29	7.3E+01	6.8E+01	6.8E+01	6.7E+01	6.9E+01	6.8E+01	6.9E+01
30	1.0E+02	1.0E+02	1.1E+02	1.1E+02	1.3E+02	1.0E+02	1.1E+02
31	1.1E+02	1.0E+02	1.1E+02	1.1E+02	1.0E+02	1.1E+02	1.1E+02
32	1.1E+02	1.1E+02	1.1E+02	1.0E+02	1.0E+02	1.0E+02	1.1E+02
33	1.1E+02	9.7E+01	1.0E+02	1.0E+02	9.9E+01	1.0E+02	1.0E+02
34	1.0E+02	9.5E+01	9.5E+01	9.2E+01	1.0E+02	9.2E+01	9.6E+01

based MLP models. The results of comparing the proposed EGOAMLP and other models against the SpamBase spam detection benchmark dataset are given in Table 15.

The results of the proposed EGOAMLP models and other models are calculated based on the definitions in Table 7 and Equations (27-36). The TP, TN, FN, and FP measurements are averaged over 100 iterations, and the remaining columns are derived from these basic measurements. The classification accuracy, detection rate, false alarm rate, Matthews correlation coefficient, positive predictive value, negative predictive value, sensitivity, specificity, G-mean, and f-measure of each evaluation measure is reported in the table and denoted as ACC, DR, FAR, MCC, PPV, NPV, SN, SP, GM, and F1, respectively.

The most important indicators are the classification accuracy, the detection rate, and the false alarm rate. The last three columns in the table (Rank accuracy, Rank Detection rate, and rank false alarm rate) highlight the rank of each algorithm according to these three indicators; the smaller the better, and denoted as RACC, RDR, and RFAR, respectively. According to the obtained results, it can be noticed that the newly proposed models achieve the highest ratios in all measurements, and they are of the top performing MLP trainers. In addition, it can be observed that the results of the BBO, DA, and MFO are better than those of other models in the majority of the measurements.

Table 15 demonstrates the results of the experiment, which were carried out using the SpamBase dataset. Spam detection model E2GOAMLP was capable of achieving the highest ratios in the three criteria: ACC, DR, and FAR with values of 96.9%, 97.2%, and 0.037, respectively. E2GOAMLP was also ranked the first best with respect to the ACC, DR, and FAR. The E4GOAMLP model was ranked the second best with respect to ACC and FAR with a value of 94.3% and 0.044 respectively, ranked fourth with respect to DR of 93.5%. The GOAMLP model was quite close to E4GOAMLP with an accuracy of 94.1%, the detection rate of 94.0%, and the false alarm rate of 0.057. GOAMLP was ranked the third best with respect to the ACC and FAR, but ranked the second-best with respect to the DR. The E5GOAMLP model is ranked fourth with respect to the ACC of 93.7% and ranked third with respect to DR of 93.8%,

TABLE 15. The measurements of the performance of 21 models to detect spam in the spambase dataset.

No.	MODEL	ACC	DR	FAR	MCC	PPV	NPV	SN	SP	F1	GM	RACC	RDR	RFAR
1	ABCMLP	73.4	81.6	0.392	0.43	0.76	0.68	0.82	0.61	0.79	70.5	19	16	21
2	ALOMLP	90.1	90.0	0.097	0.80	0.93	0.85	0.90	0.90	0.92	90.1	10	10	11
3	BBOMLP	91.2	90.2	0.072	0.82	0.95	0.86	0.90	0.93	0.93	91.5	8	9	7
4	CSMLP	88.0	88.6	0.131	0.75	0.91	0.83	0.89	0.87	0.90	87.8	12	11	13
5	DAMLP	92.5	91.7	0.063	0.85	0.96	0.88	0.92	0.94	0.94	92.7	7	7	4
6	DEMLP	82.5	80.6	0.147	0.65	0.89	0.74	0.81	0.85	0.85	82.9	15	18	14
7	GOAMLP	94.1	94.0	0.057	0.88	0.96	0.91	0.94	0.94	0.95	94.2	3	2	3
8	GSAMLP	82.0	82.4	0.186	0.63	0.87	0.75	0.82	0.81	0.85	81.9	16	15	18
9	HSMLP	71.9	71.3	0.272	0.43	0.80	0.62	0.71	0.73	0.75	72.0	20	20	19
10	MBOMLP	81.4	81.1	0.180	0.62	0.87	0.74	0.81	0.82	0.84	81.5	17	17	17
11	MFOMLP	91.2	91.5	0.092	0.82	0.94	0.87	0.92	0.91	0.93	91.2	8	8	10
12	PBILMLP	65.8	62.3	0.289	0.33	0.77	0.55	0.62	0.71	0.69	66.6	21	21	20
13	PSOMLP	81.2	79.2	0.156	0.62	0.89	0.73	0.79	0.84	0.84	81.7	18	19	15
14	SCAMLP	87.4	86.6	0.114	0.74	0.92	0.81	0.87	0.89	0.89	87.6	13	14	12
15	WOAMLP	86.2	87.4	0.156	0.71	0.90	0.81	0.87	0.84	0.88	85.9	14	13	15
16	E1GOAMLP	93.2	93.1	0.066	0.86	0.96	0.90	0.93	0.93	0.94	93.2	5	6	6
17	E2GOAMLP	96.9	97.2	0.037	0.93	0.98	0.96	0.97	0.96	0.97	96.8	1	1	1
18	E3GOAMLP	93.2	93.4	0.072	0.86	0.95	0.90	0.93	0.93	0.94	93.1	5	5	7
19	E4GOAMLP	94.3	93.5	0.044	0.88	0.97	0.91	0.94	0.96	0.95	94.6	2	4	2
20	E5GOAMLP	93.7	93.8	0.064	0.87	0.96	0.91	0.94	0.94	0.95	93.7	4	3	5
21	E6GOAMLP	89.6	87.6	0.074	0.79	0.95	0.83	0.88	0.93	0.91	90.1	11	12	9

ranked 5th with respect to FAR of 0.064. The E3GOAMLP model was ranked fifth with respect to ACC and DR with a value of 93.2% and 93.4% respectively and ranked seventh with respect to FAR of 0.072. The E1GOAMLP was also ranked the 5th with respect to the ACC of 93.2% but ranked the 6th with respect to the DR and FAR of 93.1% and 0.066, respectively.

The DAMLP model was ranked seventh with respect to ACC and DR with a value of 92.5% and 91.7% respectively, ranked fourth with respect to FAR of 0.063. The MFOMLP model was ranked 8th with respect to ACC and DR with a value of 91.4%, 91.5% respectively, and ranked 10th with respect to FAR of 0.092. Additionally, the BBOMLP model is ranked 8th with respect to ACC, 9th with respect to DR, and ranked 7th with respect to FAR with values of 91.2%, 90.2%, and 0.072, respectively. Conversely, two models performed poorly in the SpamBase dataset: HSMLP model with an inferior ACC of 71.9% and DR of 71.3%, followed by the PBILMLP model with an inferior ACC of 65.8% and DR of 62.3%. Also, in terms of FAR two models performed poorly in the SpamBase dataset: PBILMLP and ABCMLP with an inferior performance in the FAR with a value of 0.289 and 0.392, respectively.

Figures 9 and 10 demonstrate the results of the proposed models (EGOAMLP) and other metaheuristic based MLP models when applied to the SpamBase dataset, showing both convergence speed and final optimization result in terms of MSE. Investigating the convergence curves in Figure 9, we can clearly notice that the proposed models (EGOAMLP) are significantly outperforming the GOAMLP model in terms of the convergence speed, which indicates the stability of the proposed training model. Both GOAMLP and EGOAMLP

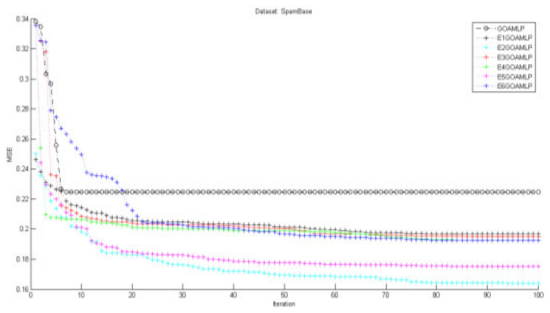

FIGURE 9. Convergence curve of GOAML and proposed model based on averages of MSE the spambase datset.

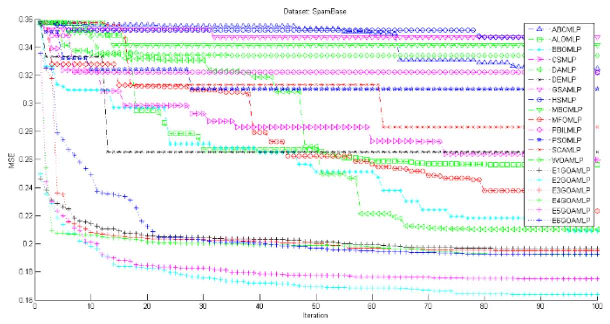

FIGURE 10. Convergence curve of all models based on averages of MSE for the spambase dataset.

models converge sharply at the early iterations; nevertheless, soon the GOAMLP almost gets stuck in the local minimum before the 7th iteration, and the global minimum decreases slightly.

On the other hand, EGOAMLP models have the strongest performance for finding a better solution and converge to a better minimum than the GOAMLP model. For example, Figure 9 shows that all the models start the optimization process of the SpamBase dataset at the same fitness, but after iteration 7, the GOAMLP model gets trapped in a suboptimal solution, while all the proposed models and specially E2GOAMLP and E5GOAMLP have a much more stable convergence rate and can find the best global function values, except for the E6GOAMLP model that has an outstanding performance after iteration 20.

As shown in Figure 10, we can clearly notice that all the proposed models and the other models begin the optimization process at the same cost. However, the proposed models are more stable and have a fast convergence rate than other models. Furthermore, all the proposed models (EGOAMLP) are capable to find the minimum values during the training over the whole optimization process.

From Figure 10, E2GOAMLP and E5GOAMLP have a little difference in final fitness cost; however, the convergence speed of E2GOAMLP is always faster than E5GOAMLP and other metaheuristic based MLP models. Thus, the MSE obtained by E2GOAMLP is better than the other models during the whole convergence process. We conclude that E2GOAMLP significantly outperforms other models in terms of convergence speed and better performance, which indicates the robustness and stability of E2GOAMLP.

Due to the limited space, Figure 11 illustrates the confusion matrices for the new proposed models in addition to some other models. It should be noted that their choice was random, and the goal was to demonstrate the performing trainers against the SpamBase dataset.

E. SCENARIO 5: PERFORMANCE OF EGOAMLP AGAINST THE SpamAssassin DATASET

This section introduces the experimental test of scenario 5 to compare the effectiveness of the proposed EGOAMLP models with 15 metaheuristic based MLP models against the SpamAssassin spam detection benchmark dataset. The results are given in Table 16. The proposed EGOAMLP models show the highest ratios in all measurement criteria (ACC, DR, FAR, MCC, PPV, NPV, SN, SP, GM, and F1) except for the last proposed model, E6GOAMLP. According to the attained results, we can notice that the proposed models can achieve the best values with respect to ACC, DR, and FAR. E2GOAMLP ranked the top, scoring 98.1%, 97.8%, and 0.012, respectively. In addition, it is very competitive in terms of GM measurement criteria with a value of 98.3%. The advantage of the GM measurement is that it indicates a good balance between SN and SP. The second rank in this dataset in terms of ACC and FAR is recorded by E3GOAMLP with values of 96.1 % and 0.014, respectively, but the second-best DR is recorded by E5GOAMLP with a value of 95.2%. The E5GOAMLP model was ranked third with respect to ACC with a score of 95.5%, and in terms of DR is scored by E4GOAMLP with a value of 95.0%, while in terms of FAR is

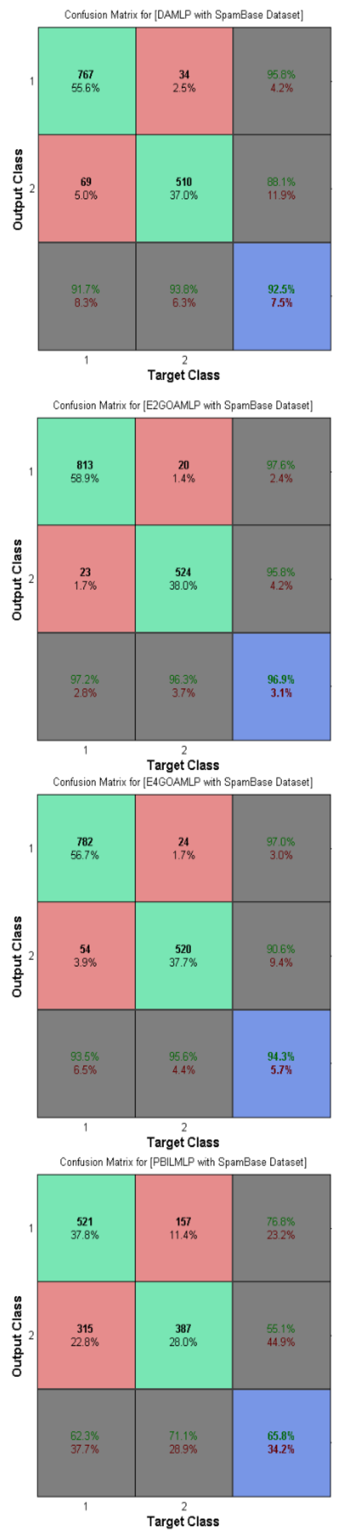

FIGURE 11. The confusion matrices against the spambase dataset.

scored by GOAMLP with values of 0.033. The E4GOAMLP model was ranked fourth with respect to ACC with a score of 94.9%, and in terms of FAR is scored by E5GOAMLP with a value of 0.040. As for the fifth rank in terms of ACC is scored

TABLE 16. The measurements of the performance of 21 models to detect spam in the spamassassin dataset.

No.	MODEL.	ACC	DR	FAR	MCC	PPV	NPV	SN	SP	F1	GM	RACC	RDR	RFAR
1	ABCMLP	79.6	75.1	0.111	0.60	0.93	0.63	0.75	0.89	0.83	81.7	21	21	20
2	ALOMLP	92.3	91.2	0.054	0.83	0.97	0.84	0.91	0.95	0.94	92.9	7	8	7
3	BBOMLP	93.4	93.4	0.067	0.85	0.97	0.87	0.93	0.93	0.95	93.4	6	5	11
4	CSMLP	91.4	90.7	0.070	0.81	0.96	0.83	0.91	0.93	0.93	91.8	9	9	12
5	DAMLP	90.6	89.9	0.081	0.80	0.96	0.82	0.90	0.92	0.93	90.9	13	11	15
6	DEMLP	85.3	83.6	0.112	0.69	0.94	0.72	0.84	0.89	0.88	86.1	20	19	21
7	GOAMLP	94.2	93.1	0.033	0.88	0.98	0.87	0.93	0.97	0.96	94.9	5	6	3
8	GSAMLP	90.3	89.6	0.083	0.79	0.96	0.81	0.90	0.92	0.93	90.6	14	13	17
9	HSMLP	87.2	83.5	0.051	0.75	0.97	0.74	0.84	0.95	0.90	89.0	19	20	5
10	MBOMLP	89.0	86.2	0.053	0.77	0.97	0.77	0.86	0.95	0.91	90.3	16	16	6
11	MFOMLP	91.1	90.4	0.076	0.81	0.96	0.82	0.90	0.92	0.93	91.4	11	10	13
12	PBILMLP	87.7	85.6	0.079	0.74	0.96	0.76	0.86	0.92	0.90	88.8	17	18	14
13	PSOMLP	90.6	88.9	0.058	0.80	0.97	0.80	0.89	0.94	0.93	91.5	12	15	9
14	SCAMLP	91.2	89.8	0.060	0.81	0.97	0.82	0.90	0.94	0.93	91.9	10	12	10
15	WOAMLP	87.6	86.2	0.095	0.74	0.95	0.76	0.86	0.91	0.90	88.3	18	16	18
16	E1GOAMLP	92.1	92.1	0.081	0.82	0.96	0.85	0.92	0.92	0.94	92.0	8	7	15
17	E2GOAMLP	98.1	97.8	0.012	0.96	0.99	0.96	0.98	0.99	0.99	98.3	1	1	1
18	E3GOAMLP	96.1	94.9	0.014	0.92	0.99	0.90	0.95	0.99	0.97	96.7	2	4	2
19	E4GOAMLP	94.9	95.0	0.054	0.89	0.97	0.90	0.95	0.95	0.96	94.8	4	3	7
20	E5GOAMLP	95.5	95.2	0.040	0.90	0.98	0.91	0.95	0.96	0.97	95.6	3	2	4
21	E6GOAMLP	89.5	89.1	0.097	0.77	0.95	0.80	0.89	0.90	0.92	89.7	15	14	19

by GOAMLP with a value of 94.2% and in terms of DR is scored by model BBOMLP with a value of 93.4%, while in terms of FAR is scored by HSMLP with values of 0.051.

The GOAMLP was ranked 6th with respect to DR, 3rd with respect to FAR with values of 93.1% and 0.033, respectively. From the previous results, we notice the superiority of all the proposed models with the SpamAssassin dataset, which were powerful and efficient models for achieving high performance of the classification accuracy with a low false alarm rate, except for the E6GOAMLP model which achieved unsatisfactory results.

The ALOMLP model is ranked 7th with respect to ACC and FAR with values of 92.3% and 0.054, respectively. The HSMLP and MBOMLP were superior to ALOMLP in terms of FAR with values of 0.051 and 0.053, respectively. The CSMLP was ranked 9th with respect to ACC and DR with values of 91.4%, and 90.7% respectively, and 12th with respect to FAR with values of 0.070, followed by the SCAMLP and MFOMLP models. SCAMLP was ranked 10th with respect to ACC and FAR with a score of 91.2% 0.060 respectively, but 12th with respect to DR, with values of 89.8%. The MFOMLP was ranked 11th with respect to ACC with a score of 91.1%, but 10th with respect to DR, and 13th with respect to FAR with values of 90.4% and 0.076, respectively.

On the other hand, the ABCMLP performed relatively poorly with this dataset in comparison to the other algorithms with an accuracy of 79.6%, and a detection rate of 75.1%. Additionally, the ABCMLP model has the best specificity of 0.89; however, it performed poorly in terms of sensitivity, which is only 0.75, which was the reason for the poor performance in terms of GM with the value of 81.7. The worst models with the SpamAssassin dataset in terms of FAR were ABCMLP and DEMLP scoring 0.111 and 0.112, respectively.

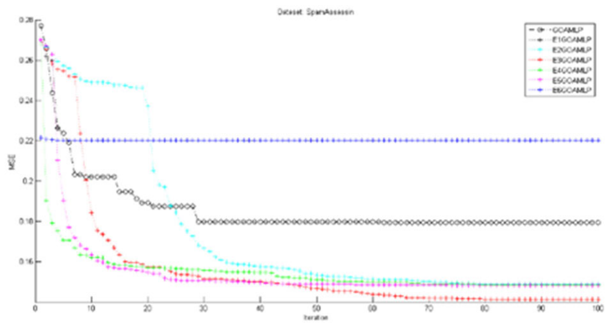

FIGURE 12. Convergence curve of goamlp and proposed models based on averages of MSE for the spamassassin dataset.

From the above analyses about the results of Table 16, we conclude the stability of the proposed models in terms of (ACC, DR, FAR) compared to the rest of the models. In addition, most of the proposed models have better performance than other models for the SpamAssassin dataset.

Moreover, the optimization process of each model is given in Figures 12 and 13. The convergence trends of the proposed models (EGOAMLP) and other metaheuristic based MLP models applied to the SpamAssassin dataset in terms of MSE are evidenced in detail. Inspecting the convergence speed and final optimization result of the curves, it can be obviously noticed that all EGOAMLP models are able to overcome other models in terms of the convergence speed and final optimization result, except for the E6GOAMLP model, where its performance was disappointing.

From Figure 12, it can be clearly noticed that E1GOAMLP, E2GOAMLP, E3GOAMLP, E4GOAMLP, and E5GOAMLP are capable of outperforming the original GOAMLP and the E6GOAMLP model in terms of the fastest convergence

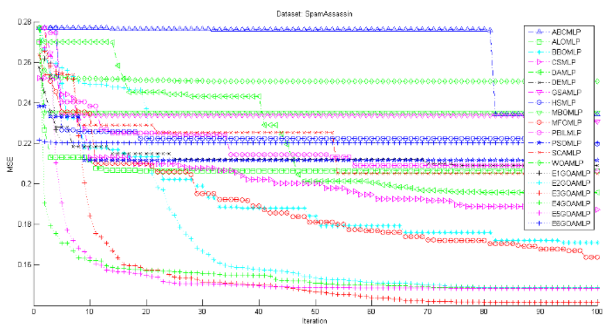

FIGURE 13. Convergence curve of all models based on averages of MSE for the spamassassin dataset.

rate and finding the global minimum. Looking carefully at Figure 12, all the models can be divided into 3 groups: the first group including E1GOAMLP, E2GOAMLP, E3GOAMLP, E4GOAMLP, and E5GOAMLP is performing the best, the second group including the original GOA is performing the second best; and the last group including E6GOAMLP is performing the worst.

Figure 13 shows the optimization process for all models against the SpamAssassin dataset. In this case, the figure obviously shows that the performance of E1GOAMLP, E2GOAMLP, E3GOAMLP, E4GOAMLP, and E5GOAMLP differs from the other models, they are superior to the rest of the models clearly, whereas E6GOAMLP fails to find the best solution in this dataset.

The BBOMLP and ALOMLP models perform the second-best in this dataset. Furthermore, all the models have the almost the same initial values, while E5GOAMLP overtakes all the other models. All models obviously outperform the ABCMLP model.

Considering the results demonstrated in Figures 12 and 13, it can be concluded that all the proposed models' performance are superior to other models and quite competitive with each other. Additionally, the results illustrate that the ALOMLP model performs much better than the other models initially, while later it is overtaken by BBOMLP, CSMLP, and MFOMLP models.

Figure 14 demonstrates the confusion matrices for the E6GOAMLP model in addition to some other models. It should be noted that their choice was random, and the goal was to show the performing trainers against the SpamAssassin dataset.

F. SCENARIO 6: PERFORMANCE OF EGOAMLP AGAINST THE UK-2011 WEBSPAM DATASET

Finally, this section introduces the numerical performance measurements and their visual representation for the most recent UK-2011 Webspam detection benchmark dataset. A sample of convergence curves and four confusion matrices are also given for our proposed models in addition to the worst and the best performance models.

These results are shown in Table 17, Figure 15, Figure 16, and Figure 17, respectively. Confirming of all previous results, the E2GOAMLP is the top-performing model in

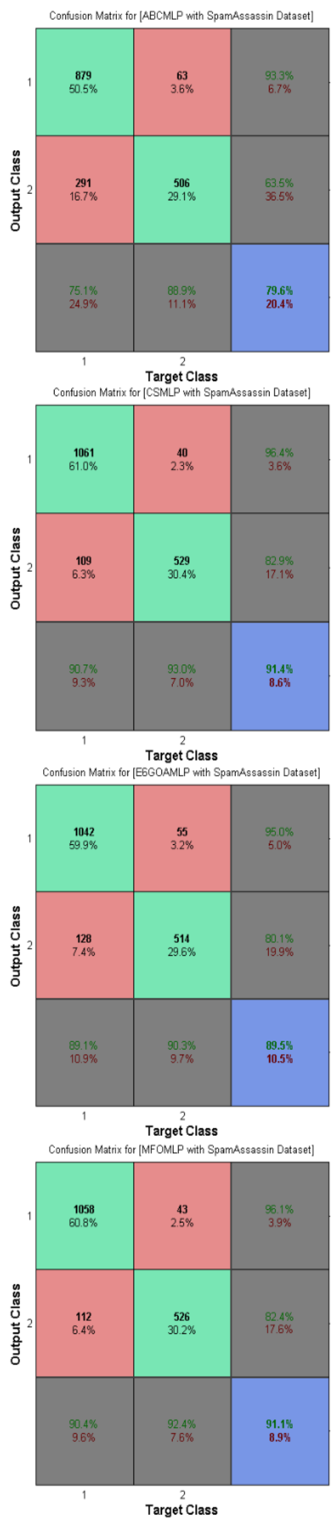

FIGURE 14. The confusion matrices against the spamassassin dataset.

this dataset too. Looking at the three last columns of Table 17 showing the ranks per ACC, DR, and FAR, the E2GOAMLP is ranked first in ACC and DR, but 2nd with respect to FAR at scores of 95.6%, 96.6%, and 0.053, respectively. E4GOAMLP was ranked 1st with respect to

TABLE 17. The measurements of the performance of 21 models to detect spam in the UK-2011webspam dataset.

No.	MODEL.	ACC	DR	FAR	MCC	PPV	NPV	SN	SP	F1	GM	RACC	RDR	RFAR
1	ABCMLP	84.3	85.8	0.170	0.69	0.82	0.87	0.86	0.83	0.84	84.4	17	16	19
2	ALOMLP	87.5	89.2	0.140	0.75	0.85	0.90	0.89	0.86	0.87	87.6	13	12	16
3	BBOMLP	91.1	92.8	0.105	0.82	0.89	0.93	0.93	0.89	0.91	91.1	8	6	11
4	CSMLP	86.1	86.8	0.145	0.72	0.84	0.88	0.87	0.85	0.85	86.1	15	14	17
5	DAMLP	89.1	90.8	0.124	0.78	0.87	0.91	0.91	0.88	0.89	89.2	11	10	13
6	DEMLP	81.2	80.6	0.182	0.62	0.80	0.83	0.81	0.82	0.80	81.2	20	20	20
7	GOAMLP	92.7	93.4	0.078	0.85	0.91	0.94	0.93	0.92	0.92	92.8	5	3	5
8	GSAMLP	80.9	82.3	0.204	0.62	0.78	0.84	0.82	0.80	0.80	80.9	21	18	21
9	HSMLP	86.8	83.6	0.104	0.74	0.88	0.86	0.84	0.90	0.86	86.6	14	17	10
10	MBOMLP	88.9	86.4	0.088	0.78	0.90	0.88	0.86	0.91	0.88	88.8	12	15	6
11	MFOMLP	90.3	91.3	0.107	0.81	0.88	0.92	0.91	0.89	0.90	90.3	10	9	12
12	PBILMLP	83.8	79.6	0.125	0.67	0.85	0.83	0.80	0.87	0.82	83.5	19	21	14
13	PSOMLP	84.0	81.5	0.139	0.68	0.84	0.84	0.82	0.86	0.83	83.8	18	19	15
14	SCAMLP	85.5	87.0	0.159	0.71	0.83	0.88	0.87	0.84	0.85	85.5	16	13	18
15	WOAMLP	91.6	93.0	0.097	0.83	0.89	0.94	0.93	0.90	0.91	91.7	6	5	7
16	E1GOAMLP	91.4	92.6	0.097	0.83	0.89	0.93	0.93	0.90	0.91	91.5	7	7	7
17	E2GOAMLP	95.6	96.6	0.053	0.91	0.94	0.97	0.97	0.95	0.95	95.6	1	1	2
18	E3GOAMLP	93.6	93.4	0.062	0.87	0.93	0.94	0.93	0.94	0.93	93.6	3	3	3
19	E4GOAMLP	95.0	94.7	0.047	0.90	0.95	0.95	0.95	0.95	0.95	95.0	2	2	1
20	E5GOAMLP	92.8	91.9	0.063	0.86	0.93	0.93	0.92	0.94	0.92	92.8	4	8	4
21	E6GOAMLP	90.4	90.6	0.097	0.81	0.89	0.92	0.91	0.90	0.90	90.4	9	11	7

FAR with a score of 0.047, but 2nd with respect to ACC and DR with values of 95.0% and 94.7%, respectively.

The E3GOAMLP model is ranked 3rd with respect to ACC, DR, and FAR at scores of 93.6%, 93.4%, and 0.062, respectively, whereas the E5GOAMLP model is ranked 4th with respect to ACC and FAR at scores of 92.8 and 0.063, respectively, while it is ranked 8th with respect to DR at a value of 91.9%. The GOAMLP model is ranked 5th with respect to ACC and FAR at scores of 92.7% and 0.078, and 3rd with respect to DR at a score of 93.4.

The WOAMLP model is ranked 6th with respect to ACC at a score of 91.6%, 5th with respect to DR at a score of 93.0%, and 7th with respect to FAR at a score of 0.097. This is followed by our model E1GOAMLP, which is ranked 7th with respect to ACC, DR, and FAR at scores of 91.4% and 92.6%, and 0.097 respectively; then followed by BBOMLP, which is ranked 8th with respect to ACC at a score of 91.1 %, 6th with respect to DR at a score of 92.8%, and 11th with respect to FAR at a score of 0.105.

Conversely, the DEMLP and GSAMLP models perform quite poorly with the UK-2011 dataset compared to the remaining models, where DEMLP is ranked 20th with respect to ACC, DR, and FAR at scores of 81.2%, 80.6%, and 0.182, and GSAMLP is ranked 21st with respect to ACC and FAR at scores of 80.9% and 0.204. The PBILMLP model is ranked 21st with respect to DR at a score of 79.6%.

In terms of convergence speed, Figures 15 and 16 demonstrate that our proposed models have a faster convergence rate compared to the other models, except for E6GOAMLP. Figure 15 shows the averaged convergence curves of the GOAMLP and the proposed training algorithms. From this figure, it can be noticed that the E2GOAMLP and E4GOAMLP models converge the fastest and significantly outperform all other models for this dataset. In this case, E3GOAMLP and E5GOAMLP are only inferior to E2GOAMLP and E4GOAMLP and perform the second best among other models. E1GOAMLP performs also well and ranks 3rd, while GOAMLP and E6GOAMLP fail to find a satisfied solution under the given conditions.

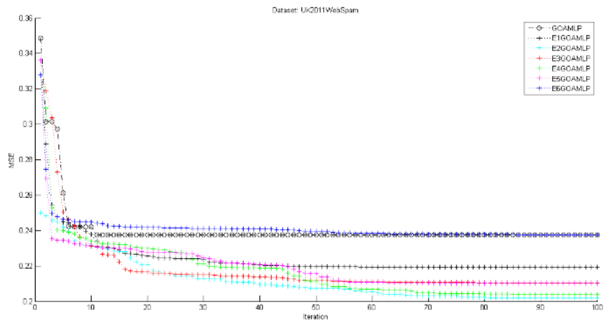

FIGURE 15. Convergence curve of all proposed models based on averages of MSE for the UK-2011webspam dataset.

Figure 16 illustrates the optimization process for all models against the UK-2011Webspam dataset. The figure shows that E1GOAMLP, E2GOAMLP, E3GOAMLP, E4GOAMLP, and E5GOAMLP performances differ from the other models, where performance is obviously superior to the rest of the models, while E6GOAMLP fail to find the best solution with this dataset. The BBOMLP and MBOMLP models perform the second-best after our proposed models (E2GOAMLP, E3GOAMLP, E4GOAMLP, and E5GOAMLP) in this dataset. Furthermore, all the models have almost the same initial values, while E2GOAMLP overtakes all the other models. All models obviously outperform the CSMLP model.

Figure 17 shows that the E2GOAMLP and E4GOAMLP models have overall superior performances in terms of accuracy, detection rate, and false alarm rate. Additionally, the E2GOAMLP model has a superior performance in terms of the confusion matrix. As illustrated in Figure 17, the E2GOAMLP model has the lowest coefficient values of

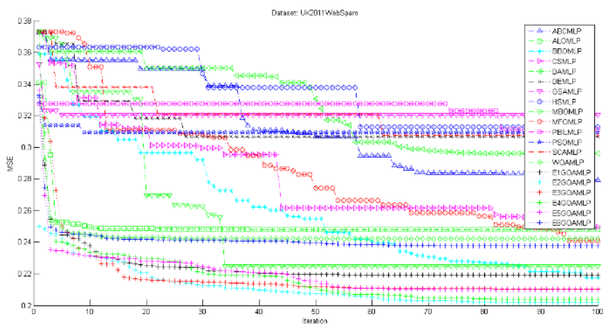

FIGURE 16. Convergence curve of all models based on averages of MSE for the UK-2011webspam dataset.

the false positives (FP) and false negatives (FN) compared to the other 20 models. On the contrary, the E2GOAMLP model has the highest coefficient value of the true positives (TP) and true negatives (TN), where the correctly predicted cases are located along with the matrix diagonal.

Figure 17 illustrates the confusion matrices for the ABCMLP model in addition to MBOMLP, E4GOAMLP, and E2GOAMLP. It should be noted that their choice was random, and the goal was to demonstrate the performing trainers against the UK-2011Webspam dataset.

1) PERFORMANCE COMPARISON OF PROPOSED MODELS AND OTHER SCHEMES

In this section, we have compared the performance of the proposed model with the more recent proposed schemes of spam detection systems from state-of -arts are listed in Table 18.

This comparison demonstrates the contribution and superiority of our model on publicly available datasets including the SpamBase, SpamAssassin, and UK-2011 Webspam datasets. The proposed model has the best performance in the evaluation criteria. The data were correctly classified by the proposed model compared to those classified by the static schemes. Moreover, GOAMLP showed a significantly lower FAR than some of the recent schemes.

2) PERFORMANCE EVALUATION USING T-TEST

In this section, we have analyzed statistically the performance of the proposed algorithm of the previous results, the tests were performed in two phases: In the first phase, to be able to correctly judge the statistical analysis of the previous results, we have conducted the statistical t-test (T) in order to evaluate the real performance of the proposed algorithms (EGOAs) as opposed to the original algorithm (GOA) on 34 benchmark functions based on the two-tailed test to reveal the differences in the obtained best function minima are statically significant. When comparing two algorithms, algorithm 1 and algorithm 2, the null hypothesis H0 is defined as the claim that algorithm 1 and algorithm 2 performed equally well. The alternative hypothesis H1 is defined as the assumption that algorithm 1 overcame algorithm 2. Algorithm 1 is EGOA in all the tests. The significance level of the p-value was set to 0.05, i.e., the alternative hypothesis H1 would be accepted if the p-value were less than 0.05 (i.e., 95% confidence level).

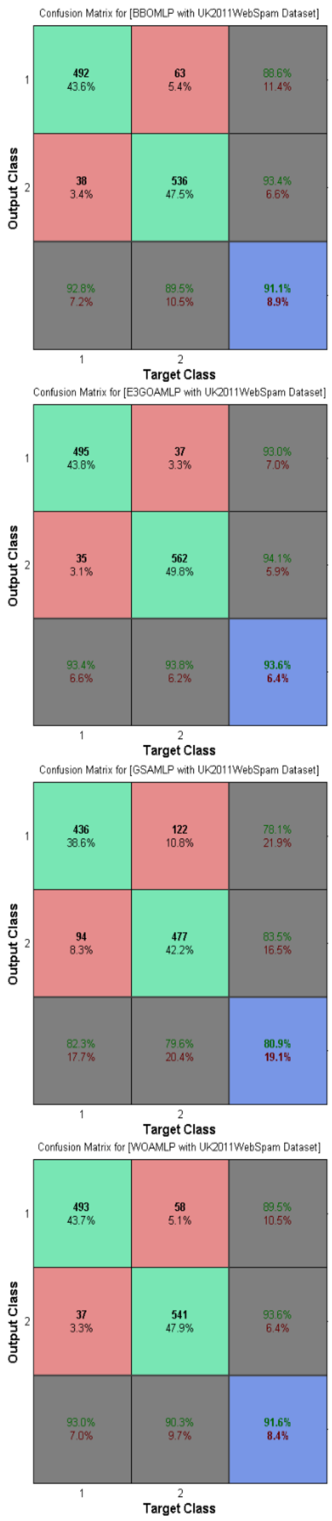

FIGURE 17. The confusion matrices against the UK-2011webspam dataset.

Table 19 presents the p-values from the paired t-tests between the proposed algorithms (EGOAs) and the original algorithm (GOA). The boldface demonstrates that the performance of the original algorithm (GOA) is better than the proposed algorithms (EGOAs). The last three rows of this table represent that proposed algorithms (EGOAs) are better than,

TABLE 18. Comparison results with other schemes.

No.	Ref.	Year	Dataset	Method	EC	Results
1	[72]	2012	UK-2011	SEO	ACC	89.01
2	[73]	2016	UK-2011	MLP-GD	DR	82.39
3	[74]	2012	UK-2011	D.F.	ACC	95.05
4	[57]	2017	UK-2011	EBATFFNN	N/A	N/A
5	[75]	2016	SpamAssassin	S-1D-LBP	ACC	93.04
6	[76]	2014	SpamAssassin	SVM	ACC	91.47
				PSOSVM	ACC	93.19
7	[77]	2017	SpamAssassin	GA-SVM	ACC	91
				GA-ALS	ACC	95
				SVM	ACC	88
8	[55]	2015	SpamAssassin	KHFFNN	ACC	91.08
9	[78]	2019	SpamAssassin	GARWN	ACC	92.2
10	[58]	2020	SpamAssassin	MLBCA-SR	ACC	77.51
11	[60]	2020	SpamAssassin	FSEDM	ACC	97.2
12	[79]	2020	SpamBase	WOAFPA	ACC	94
13	[80]	2019	SpamBase	SSA-KNN	ACC	94
14	[81]	2018	SpamBase	MLP	ACC	93
15	[82]	2020	SpamBase	SVM	ACC	89.21
				RF	ACC	91.36
16	[83]	2020	SpamBase	SCAC	ACC	94
17	[5]	2019	SpamBase	SVM	ACC	94.06
18	[4]	2019	SpamBase	WCASASVM	ACC	94
				WCASANB		84
				WCASAKNN		87
19	[51]	2015	SpamBase	NSA–PSO	ACC	83.20
20	[3]	2014	SpamBase	NSA–PSO	ACC	91.22
				PSO		81.32
				NSA		68.86
21	[59]	2021	SpamBase	BP+M	ACC	95.38
22	Our SDS		UK-2011			95.6
			SpamAssassin			98.1
			SpamBase			96.9

EC: Evaluation Criteria

equal to, and worse than the original algorithm (GOA). For instance, the comparison between E3GOA and GOA indicates that E3GOA performs better than, equal to, and worse than GOA on 54, 1, and 5 functions, respectively. Similarly, we can say that E2GOA performs better and worse than GOA on 52 and 3 functions, respectively. E1GOA performs better than, equal to and worse than GOA on 46, 11, and 3 benchmark functions. Table 19 shows that all the proposed algorithms (E1GOA, E2GOA, E3GOA, E4GOA, E5GOA, and E6GOA) outperform the original algorithm (GOA) on most of the benchmark functions.

In the second phase, the Statistical t-test for EGOAMLP against the other models. In comparing two models, model 1 and model 2, the null hypothesis H0 is defined as the claim that model 1 and model 2 performed equally well. The alternative hypothesis H1 is defined as the assumption that model 1 overcame model 2. Model 1 is one of the proposed models (EGOAMLP) in all the tests. The significance level of the p-value was set to 0.05, i.e., the alternative hypothesis H1 would be accepted if the p-value were less than 0.05 (i.e., 95% confidence level). Table 20 presents the p-values from the paired t-tests between proposed models (E1GOAMLP, E2GOAMLP, E3GOAMLP, E4GOAMLP, E5GOAMLP, and E6GOAMLP) and each of ABC, ALO, BBO, DA, CS, DE, GOA, GSA, MBO, HS, MFO, PBIL, PSO, SCA, and WOA.

TABLE 19. Comparison between EGOAS and GOA at A = 0.05 on a two-tailed t-test.

Fun.	E1GOA	E2GOA	E3GOA	E4GOA	E5GOA	E6GOA
1	2.53E-05	2.65E-05	2.79E-05	1.38E-05	8.42E-06	4.75E-06
2	7.00E-65	7.58E-05	5.97E-76	3.45E-55	1.16E-144	4.03E-69
3	7.15E-08	8.91E-11	6.13E-13	1.14E-12	2.71E-11	8.09E-13
4	8.86E-34	8.39E-32	1.72E-35	2.43E-34	4.04E-36	4.63E-37
5	5.70E-03	7.71E-03	1.06E-02	3.77E-02	5.95E-03	3.14E-02
6	3.71E-05	5.62E-05	2.50E-05	5.58E-04	3.30E-05	7.57E-05
7	1.23E-09	3.84E-09	6.26E-10	1.05E-09	2.89E-10	2.82E-09
8	9.11E-67	3.71E-08	9.06E-44	7.96E-59	**4.59E-01**	7.06E-67
9	2.69E-52	3.24E-268	5.99E-61	1.81E-42	5.96E-55	1.77E-76
10	1.45E-90	0.00E+00	2.02E-84	3.86E-68	1.14E-85	4.85E-64
11	2.45E-05	1.82E-05	2.04E-05	2.53E-04	1.25E-05	3.30E-05
12	2.71E-02	2.88E-02	**8.69E-02**	4.74E-02	2.88E-02	**9.36E-02**
13	1.73E-02	1.73E-02	3.15E-02	1.59E-02	2.79E-02	**7.44E-02**
14	-	3.86E-05	1.16E-24	1.45E-04	2.60E-14	**4.69E-01**
15	**4.24E-01**	4.14E-01	3.12E-06	**8.03E-01**	6.99E-06	6.56E-14
16	-	1.08E-04	5.29E-05	**6.64E-01**	2.08E-04	3.09E-09
17	-	1.74E-11	1.91E-04	1.50E-02	4.85E-03	3.27E-02
18	-	-	-	-	-	-
19	1.55E-110	-	3.16E-157	7.55E-102	-	4.15E-85
20	1.62E-98	-	3.08E-122	1.59E-88	-	5.55E-72
21	1.15E-47	1.71E-09	8.07E-54	2.86E-59	2.21E-11	9.40E-60
22	9.59E-06	1.05E-11	4.16E-18	2.30E-17	2.68E-20	2.96E-18
23	1.19E-13	9.54E-08	3.68E-15	5.52E-15	7.24E-05	4.15E-15
24	1.29E-06	1.29E-05	2.39E-07	1.37E-05	8.98E-06	7.35E-06
25	9.00E-03	7.43E-03	**9.96E-02**	2.78E-02	6.86E-03	1.25E-02
26	2.80E-52	3.89E-48	1.15E-50	2.42E-51	1.75E-48	1.36E-50
27	1.09E-91	2.19E-39	4.08E-84	1.31E-88	2.84E-61	2.18E-72
28	1.44E-03	1.58E-03	1.51E-03	1.11E-03	3.83E-04	1.59E-03
29	1.50E-18	5.89E-20	1.15E-05	9.49E-06	1.47E-02	5.36E-17
30	4.47E-43	3.18E-93	1.94E-50	6.46E-38	1.20E-56	4.56E-23
31	1.21E-05	2.87E-05	2.21E-05	1.45E-05	3.14E-05	1.32E-08
32	1.58E-48	1.78E-06	3.17E-50	5.68E-51	9.26E-06	4.65E-49
33	1.13E-10	4.78E-09	1.80E-09	3.42E-10	6.64E-10	1.54E-09
34	5.73E-05	1.98E-03	5.97E-05	6.76E-05	6.95E-05	2.50E-05
Better	29	31	31	31	30	30
Equal	4	3	1	1	3	1
worse	1	0	2	2	1	3

(-) No difference

The p value (probability) is used to determine if the MSE mean differs or not. All the p-values were smaller than 0.05, except for the tests between DAMLP and E1GOAMLP and between DAMLP and E6GOAMLP with the SpamAssassin dataset. Also, the differences between E6GOAMLP and PBILMLP and between E6GOAMLP and SCAMLP were not statistically significant in the SpamAssassin dataset. All the proposed models were always better than the other models with the SpamBase dataset. Finally, the differences between DAMLP and E1GOAMLP, E3GOAMLP, and E5GOAMLP were not statistically significant with UK-2011 Webspam.

The same is also true for the difference between GOAMLP and E6GOAMLP. The p-values greater than 0.05 are shown in boldface to indicate that the hypothesis of EGOAMLP superiority cannot be accepted. Therefore, there is a significant correlation between the mean of the proposed models of differential evolution. This also shows a high level of accuracy between them. The results below show an obvious

TABLE 20. P-value for EGOAMLP against the other models.

Dataset	*Model*	*E1*	*E2*	*E3*	*E4*	*E5*	*E6*
SpamAssassin	ABC	4.E-47	7.E-44	4.E-59	5.E-80	3.E-68	2.E-50
	ALO	1.E-52	1.E-13	6.E-35	2.E-87	5.E-53	4.E-34
	BBO	4.E-28	5.E-12	4.E-44	8.E-44	3.E-45	4.E-22
	CS	2.E-26	3.E-16	4.E-48	1.E-66	2.E-58	2.E-19
	DA	**9.E-01**	1.E-39	1.E-50	2.E-50	3.E-47	**6.E-01**
	DE	5.E-37	2.E-18	2.E-41	3.E-90	2.E-61	4.E-16
	GOA	6.E-51	1.E-08	7.E-36	2.E-66	2.E-56	2.E-36
	GSA	1.E-79	4.E-29	2.E-51	8.E-97	8.E-75	3.E-38
	HS	9.E-09	3.E-25	2.E-50	7.E-92	7.E-79	2.E-05
	MBO	2.E-43	8.E-28	3.E-47	6.E-91	9.E-64	1.E-60
	MFO	4.E-28	5.E-10	9.E-42	2.E-39	5.E-41	2.E-22
	PBIL	7.E-03	3.E-27	8.E-57	3.E-86	1.E-78	**3.E-01**
	PSO	7.E-26	1.E-17	3.E-41	3.E-78	1.E-57	1.E-18
	SCA	1.E-02	6.E-26	7.E-52	4.E-78	8.E-63	**2.E-01**
	WOA	1.E-73	1.E-35	2.E-54	3.E-95	6.E-71	1.E-109
SpamBase	ABC	1.E-114	1.E-109	1.E-82	7.E-97	7.E-94	1.E-74
	ALO	5.E-48	5.E-69	4.E-46	3.E-46	2.E-60	3.E-66
	BBO	4.E-35	4.E-58	6.E-34	1.E-35	9.E-49	1.E-40
	CS	6.E-80	1.E-113	2.E-71	1.E-73	2.E-89	3.E-78
	DA	3.E-24	1.E-37	5.E-24	9.E-25	2.E-32	1.E-25
	DE	1.E-66	1.E-92	6.E-60	9.E-62	5.E-83	4.E-68
	GOA	6.E-107	1.E-132	1.E-72	2.E-82	2.E-97	4.E-69
	GSA	6.E-131	4.E-110	9.E-85	3.E-103	3.E-97	1.E-73
	HS	6.E-123	7.E-108	2.E-84	2.E-101	1.E-95	4.E-73
	MBO	5.E-139	2.E-113	2.E-86	6.E-104	2.E-99	1.E-74
	MFO	1.E-44	1.E-63	8.E-43	3.E-44	5.E-55	2.E-48
	PBIL	3.E-138	3.E-112	1.E-86	3.E-103	1.E-99	3.E-71
	PSO	5.E-145	2.E-129	2.E-86	1.E-101	8.E-104	2.E-75
	SCA	7.E-78	8.E-96	7.E-68	2.E-71	9.E-82	4.E-77
	WOA	7.E-147	2.E-118	4.E-87	2.E-103	7.E-102	5.E-76
UK 2011 Webspam	ABC	6.E-58	2.E-71	3.E-60	1.E-71	8.E-66	5.E-47
	ALO	3.E-62	3.E-52	6.E-54	6.E-51	3.E-54	6.E-24
	BBO	4.E-43	3.E-32	7.E-39	7.E-46	1.E-38	2.E-48
	CS	6.E-45	9.E-58	2.E-52	3.E-59	2.E-50	4.E-28
	DA	**2.E-01**	3.E-07	**2.E-01**	2.E-02	**3.E-01**	9.E-12
	DE	4.E-98	6.E-115	1.E-100	3.E-93	8.E-90	5.E-77
	GOA	4.E-44	2.E-42	5.E-50	1.E-35	1.E-37	**8.E-01**
	GSA	1.E-90	1.E-99	2.E-76	3.E-79	9.E-91	9.E-108
	HS	5.E-78	4.E-99	4.E-76	6.E-90	2.E-87	4.E-70
	MBO	5.E-11	4.E-20	3.E-16	1.E-17	4.E-14	7.E-03
	MFO	1.E-42	4.E-31	9.E-39	2.E-47	1.E-38	1.E-47
	PBIL	4.E-84	4.E-102	3.E-74	4.E-79	3.E-87	3.E-91
	PSO	8.E-85	7.E-97	7.E-72	3.E-74	9.E-85	7.E-98
	SCA	3.E-100	2.E-127	1.E-93	4.E-105	1.E-104	2.E-88
	WOA	5.E-55	1.E-43	4.E-38	4.E-41	7.E-54	9.E-11

difference between the performance of our proposed models (EGOAMLP) and the rest of the models. In this table, p-values of less than 10^{-2}, and those of smaller than 0.05 are shown in regular face to indicate that the hypothesis of our proposed models (EGOAMLP) superiority can be accepted.

VI. CONCLUSION

This research introduced a series of six different variants of the models for the spam detection system, namely, the EGOAMLPs. The six different models have been investigated on three spam datasets: SpamAssassin, SpamBase, and UK-2011Webspam. Generally speaking, the paper focuses on the applicability of six different variants of the EGOAMLP model to train MLP in order to develop a new spam detection system (SDS). The confusion matrix is the main source of the basic measurements of TP, TN, FN, and FP. The performance of the six proposed variants was compared with a number of recent spam detection systems. The current study utilized 15 of the metaheuristic algorithms to train the MLP such as ALO, ABC, BBO, DA, CS, DE, GOA, GSA, HS, MFO, MBO, PBIL, SCA, PSO, and WOA. By looking to the obtained results, it can be observed that the proposed models (namely, E1GOAMLP, E2GOAMLP, E3GOAMLP, E4GOAMLP, and E5GOAMLP) are able to significantly improve the performance of classification. The reason for the improved performance is that the proposed models provide a better balance between exploration and exploitation that prevents the models from stagnating in local optima during the optimization process. The results also indicate that most EGOAMLPs decrease the comfort zone, attraction zone, and repulsion zone between grasshoppers in a much better way which allows the EGOAMLP models to show better capability in terms of ACC, DR, and FAR. The results suggest that the tuned models evidently escalate the reliability of the global optimality and classification accuracy, they also enhance the quality of the results. The results show that the best training model of all is E2GOAMLP with the SpamBase, SpamAssassin, and UK-2011Webspam datasets, having classification accuracies of 96.9%, 98.1%, and 95.6%, respectively, and detection rates of 97.2%, 97.8%, and 96.6%, respectively, and finally FAR values of 0.037, 0.012, and 0.053, respectively. However, this research only evaluated the models by using all the features of the spam detection datasets where an adequate feature selection technique has not been included. Therefore, in the future our research will focus on developing an effective SDS based on two goals: the first is to reduce the number of selected features, and the second is to hybridize among the best-proposed models to build an ensemble-based classifier for spam detection.

REFERENCES

[1] N. Hussain, H. T. Mirza, I. Hussain, F. Iqbal, and I. Memon, "Spam review detection using the linguistic and spammer behavioral methods," *IEEE Access*, vol. 8, pp. 53801–53816, 2020.

[2] A. Karim, S. Azam, B. Shanmugam, K. Kannoorpatti, and M. Alazab, "A comprehensive survey for intelligent spam email detection," *IEEE Access*, vol. 7, pp. 168261–168295, 2019.

[3] I. Idris and A. Selamat, "Improved email spam detection model with negative selection algorithm and particle swarm optimization," *Appl. Soft Comput.*, vol. 22, pp. 11–27, Sep. 2014.

[4] G. Al-Rawashdeh, R. Mamat, and N. H. B. A. Rahim, "Hybrid water cycle optimization algorithm with simulated annealing for spam E-mail detection," *IEEE Access*, vol. 7, pp. 143721–143734, 2019.

[5] S. O. Olatunji, "Improved email spam detection model based on support vector machines," *Neural Comput. Appl.*, vol. 31, no. 3, pp. 691–699, Mar. 2019.

[6] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: Review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, Jun. 2019, Art. no. e01802.

[7] O. Fonseca, E. Fazzion, I. Cunha, P. H. B. Las-Casas, D. Guedes, W. Meira, C. Hoepers, K. Steding-Jessen, and M. H. P. Chaves, "Measuring, characterizing, and avoiding spam traffic costs," *IEEE Internet Comput.*, vol. 20, no. 4, pp. 16–24, Jul. 2016.

[8] J. R. Mendez, F. Fdez-Riverola, F. Diaz, E. L. Iglesias, and J. M. Corchado, "A comparative performance study of feature selection methods for the anti-spam filtering domain," in *Proc. Ind. Conf. Data Mining*. Berlin, Germany: Springer, 2006, pp. 106–120.

[9] K. Li, Z. Zhong, and L. Ramaswamy, ''Privacy-aware collaborative spam filtering,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 5, pp. 725–739, May 2009.

[10] Z. Zhang, R. Hou, and J. Yang, ''Detection of social network spam based on improved extreme learning machine,'' *IEEE Access*, vol. 8, pp. 112003–112014, 2020.

[11] S. Gibson, B. Issac, L. Zhang, and S. M. Jacob, ''Detecting spam email with machine learning optimized with bio-inspired metaheuristic algorithms,'' *IEEE Access*, vol. 8, pp. 187914–187932, 2020.

[12] D. Liu and J.-H. Lee, ''CNN based malicious website detection by invalidating multiple web spams,'' *IEEE Access*, vol. 8, pp. 97258–97266, 2020.

[13] A. C. Pandey and D. S. Rajpoot, ''Spam review detection using spiral cuckoo search clustering method,'' *Evol. Intell.*, vol. 12, no. 2, pp. 147–164, Jun. 2019.

[14] J.-F. Tsai, M.-H. Lin, and D.-Y. Wen, ''Global optimization for mixed–discrete structural design,'' *Symmetry*, vol. 12, no. 9, p. 1529, Sep. 2020.

[15] W. A. H. M. Ghanem and A. Jantan, ''Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems,'' *Neural Comput. Appl.*, vol. 30, no. 1, pp. 163–181, Jul. 2018.

[16] B. Kizielewicz and W. Sałabun, ''A new approach to identifying a multi-criteria decision model based on stochastic optimization techniques,'' *Symmetry*, vol. 12, no. 9, p. 1551, Sep. 2020.

[17] A. Milani, ''Evolutionary algorithms in intelligent systems,'' *Mathematics*, vol. 8, no. 10, p. 1733, Oct. 2020.

[18] S. Mirjalili, ''Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems,'' *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, May 2016.

[19] X. S. Yang and S. Deb, ''Cuckoo search via Lévy flights,'' in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, 2009, pp. 210–214.

[20] S. Mirjalili, ''The ant lion optimizer,'' *Adv. Eng. Softw.*, vol. 83, pp. 80–98, May 2015.

[21] P. Soltani and E. Hadavandi, ''A monarch butterfly optimization-based neural network simulator for prediction of siro-spun yarn tenacity,'' *Soft Comput.*, vol. 23, pp. 1–15, Nov. 2018.

[22] W. A. H. M. Ghanem and A. Jantan, ''An enhanced bat algorithm with mutation operator for numerical optimization problems,'' *Neural Comput. Appl.*, vol. 31, no. S1, pp. 617–651, Jan. 2019.

[23] D. Manjarres, I. G.-L. S. Landa-Torres, S. J. Del, M. N. Bilbao, S. Salcedo-Sanz, and Z. W. Geem, ''A survey on applications of the harmony search algorithm,'' *Eng. Appl. Artif. Intell.*, vol. 26, no. 8, pp. 1818–1831, 2013.

[24] S. Mirjalili, ''Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm,'' *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.

[25] D. Wu, S. Xu, and F. Kong, ''Convergence analysis and improvement of the chicken swarm optimization algorithm,'' *IEEE Access*, vol. 4, pp. 9400–9412, 2016.

[26] S. Mirjalili, ''SCA: A sine cosine algorithm for solving optimization problems,'' *Knowl.-Based Syst.*, vol. 96, pp. 120–133, Mar. 2016.

[27] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, ''Salp swarm algorithm: A bio-inspired optimizer for engineering design problems,'' *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[28] G. I. Sayed, G. Khoriba, and M. H. Haggag, ''A novel chaotic salp swarm algorithm for global optimization and feature selection,'' *Appl. Intell.*, vol. 48, no. 10, pp. 3462–3481, 2018.

[29] K. Socha and C. Blum, ''An ant colony optimization algorithm for continuous optimization: Application to feed-forward neural network training,'' *Neural Comput. Appl.*, vol. 16, no. 3, pp. 235–247, May 2007.

[30] X. Li, J. Zhang, and M. Yin, ''Animal migration optimization: An optimization algorithm inspired by animal migration behavior,'' *Neural Comput. Appl.*, vol. 24, nos. 7–8, pp. 1867–1877, Jun. 2014.

[31] C. H. Wu, ''Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks,'' *Expert Syst. Appl.*, vol. 36, no. 3, pp. 4321–4330, Apr. 2009.

[32] A. H. Mohammad and R. A. Zitar, ''Application of genetic optimized artificial immune system and neural networks in spam detection,'' *Appl. Soft Comput.*, vol. 11, no. 4, pp. 3827–3845, Jun. 2011.

[33] M.-C. Su, H.-H. Lo, and F.-H. Hsu, ''A neural tree and its application to spam e-mail detection,'' *Expert Syst. Appl.*, vol. 37, no. 12, pp. 7976–7985, Dec. 2010.

[34] M. Zmyślony, B. Krawczyk, and M. Woźniak, ''Combined classifiers with neural fuser for spam detection,'' in *Proc. Int. Joint Conf. CISIS-ICEUTE-SOCO Special Sessions*. Berlin, Germany: Springer, 2013, pp. 245–252.

[35] K. Manjusha and R. Kumar, ''Spam mail classification using combined approach of Bayesian and neural network,'' in *Proc. Int. Conf. Comput. Intell. Commun. Netw.*, Nov. 2010, pp. 145–149.

[36] H. Xu and B. Yu, ''Automatic thesaurus construction for spam filtering using revised back propagation neural network,'' *Expert Syst. Appl.*, vol. 37, no. 1, pp. 18–23, Jan. 2010.

[37] S. Mirjalili, ''How effective is the Grey Wolf optimizer in training multi-layer perceptrons,'' *Appl. Intell.*, vol. 43, no. 1, pp. 150–161, 2015.

[38] W. A. H. M. Ghanem and A. Jantan, ''A cognitively inspired hybridization of artificial bee colony and dragonfly algorithms for training multi-layer perceptrons,'' *Cognit. Comput.*, vol. 10, no. 6, pp. 1096–1134, Dec. 2018.

[39] W. A. H. Ghanem and A. Jantan, ''Using hybrid artificial bee colony algorithm and particle swarm optimization for training feed-forward neural networks,'' *J. Theor. Appl. Inf. Technol.*, vol. 67, no. 3, pp. 1–11, 2014.

[40] W. A. H. M. Ghanem and A. Jantan, ''Training a neural network for cyberattack classification applications using hybridization of an artificial bee colony and monarch butterfly optimization,'' *Neural Process. Lett.*, vol. 51, pp. 1–42, Oct. 2020.

[41] W. A. H. M. Ghanem and A. Jantan, ''Swarm intelligence and neural network for data classification,'' in *Proc. IEEE Int. Conf. Control Syst., Comput. Eng. (ICCSCE)*, Nov. 2014, pp. 196–201.

[42] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Faris, and I. Aljarah, ''Grasshopper optimization algorithm for multi-objective optimization problems,'' *Appl. Intell.*, vol. 48, no. 4, pp. 805–820, 2018.

[43] S. A. A. Ghaleb, M. Mohamad, E. F. H. S. Abdullah, and W. A. H. M. Ghanem, ''Integrating mutation operator into grasshopper optimization algorithm for global optimization,'' *Soft Comput.*, vol. 25, pp. 1–44, Apr. 2021.

[44] A. A. Ewees, M. A. Elaziz, and E. H. Houssein, ''Improved grasshopper optimization algorithm using opposition-based learning,'' *Expert Syst. Appl.*, vol. 112, pp. 156–172, Dec. 2018.

[45] J. Luo, H. Chen, Q. Zhang, Y. Xu, H. Huang, and X. Zhao, ''An improved grasshopper optimization algorithm with application to financial stress prediction,'' *Appl. Math. Model.*, vol. 64, pp. 654–668, Dec. 2018.

[46] S. Arora and P. Anand, ''Chaotic grasshopper optimization algorithm for global optimization,'' *Neural Comput. Appl.*, vol. 31, pp. 1–21, Jan. 2018.

[47] X. Wenhan, W. Yuanxing, Q. Di, and B. D. Rouyendegh, ''Improved grasshopper optimization algorithm to solve energy consuming reduction of chiller loading,'' *Energy Sources A*, pp. 1–14, 2019, doi: 10.1080/15567036.2019.1687622.

[48] R. Zhao, H. Ni, H. Feng, Y. Song, and X. Zhu, ''An improved grasshopper optimization algorithm for task scheduling problems,'' *Int. J. Innov. Comput., Inf. Control*, vol. 15, pp. 1967–1987, Oct. 2019.

[49] P. Mishra, V. Goyal, and A. Shukla, ''An improved grasshopper optimization algorithm for solving numerical optimization problems,'' in *Advances in Intelligent Computing and Communication*. Singapore: Springer, 2020, pp. 179–188.

[50] V. K. Ojha, A. Abraham, and V. Snášel, ''Metaheuristic design of feedforward neural networks: A review of two decades of research,'' *Eng. Appl. Artif. Intell.*, vol. 60, pp. 97–116, Apr. 2017.

[51] I. Idris, A. Selamat, N. T. Nguyen, S. Omatu, O. Krejcar, K. Kuca, and M. Penhaker, ''A combined negative selection algorithm-particle swarm optimization for an email spam detection system,'' *Eng. Appl. Artif. Intell.*, vol. 39, pp. 33–44, Mar. 2015.

[52] B. K. Dedeturk and B. Akay, ''Spam filtering using a logistic regression model trained by an artificial bee colony algorithm,'' *Appl. Soft Comput.*, vol. 91, Jun. 2020, Art. no. 106229.

[53] I. Idris, ''E-mail spam classification with artificial neural network and negative selection algorithm,'' *Int. J. Comput. Sci. Commun. Netw.*, vol. 1, no. 3, pp. 227–231, 2011.

[54] S. Singh, A. Chand, and S. P. Lal, ''Improving spam detection using neural networks trained by memetic algorithm,'' in *Proc. 5th Int. Conf. Comput. Intell., Modelling Simulation*, Sep. 2013, pp. 55–60.

[55] H. Faris, I. Aljarah, and J. Alqatawna, ''Optimizing feedforward neural networks using krill herd algorithm for E-mail spam detection,'' in *Proc. IEEE Jordan Conf. Appl. Electr. Eng. Comput. Technol. (AEECT)*, Nov. 2015, pp. 1–5.

[56] A. Rodan, H. Faris, and J. Alqatawna, ''Optimizing feedforward neural networks using biogeography based optimization for E-mail spam identification,'' *Int. J. Commun., Netw. Syst. Sci.*, vol. 9, no. 1, pp. 19–28, 2016.

[57] A. Jantan, W. A. Ghanem, and S. A. Ghaleb, ''Using modified bat algorithm to train neural networks for spam detection,'' *J. Theor. Appl. Inf. Technol.*, vol. 95, no. 24, pp. 1–12, 2017.

[58] D. Davino, F. Camastra, A. Ciaramella, and A. Staiano, "Spam detection by machine learning-based content analysis," in *Progresses in Artificial Intelligence and Neural Systems*. Singapore: Springer, 2020, pp. 415–422.

[59] S. Sinha, I. Ghosh, and S. C. Satapathy, "A study for ANN model for spam classification," in *Intelligent Data Engineering and Analytics*. Singapore: Springer, 2021, pp. 331–343.

[60] A. Zamir, H. U. Khan, W. Mehmood, T. Iqbal, and A. U. Akram, "A feature-centric spam email detection model using diverse supervised machine learning algorithms," *Electron. Library*, vol. 38, no. 3, pp. 633–657, Jul. 2020.

[61] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: Theory and application," *Adv. Eng. Softw.*, vol. 105, pp. 30–47, Mar. 2017.

[62] S. Saremi, S. Mirjalili, S. Mirjalili, and J. S. Dong, "Grasshopper optimization algorithm: Theory, literature review, and application in hand posture estimation," in *Nature-Inspired Optimizers*. Cham, Switzerland: Springer, 2020, pp. 107–122.

[63] G. G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Comput. Appl.*, vol. 31, pp. 1–20, May 2015.

[64] A. A. Heidari, H. Faris, I. Aljarah, and S. Mirjalili, "An efficient hybrid multilayer perceptron neural network with grasshopper optimization," *Soft Comput.*, vol. 23, no. 17, pp. 7941–7958, 2018.

[65] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *J. Simul.*, vol. 76, no. 2, pp. 60–68, Feb. 2001.

[66] W. A. Ghanem and A. Jantan, "A new approach for intrusion detection system based on training multilayer perceptron by using enhanced Bat algorithm," *Neural Comput. Appl.*, vol. 32, pp. 1–34, Dec. 2019.

[67] W. A. H. Ghanem and A. Jantan, "New approach to improve anomaly detection using a neural network optimized by hybrid ABC and PSO algorithms," *Pakistan J. Statist.*, vol. 34, no. 1, pp. 1–14, 2018.

[68] W. A. H. M. Ghanem, A. Jantan, S. A. A. Ghaleb, and A. B. Nasser, "An efficient intrusion detection model based on hybridization of artificial bee colony and dragonfly algorithms for training multilayer perceptrons," *IEEE Access*, vol. 8, pp. 130452–130475, 2020.

[69] M. Hopkins. (1999). UCI machine learning repository: SpamBase data set. Hewlett-Packard Labs. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/SpamBase

[70] M. Hopkins *et al. UCI Machine Learning Repository: SpamAssassin Data Set*. Accessed: Nov. 29, 2020. [Online]. Available: https://www.kaggle.com/beatoa/spamassassin-public-corpus

[71] *UK-2011 Web Spam Dataset*. Accessed: Jul. 2018. [Online]. Available: https://sites.google.com/site/heiderawahsheh/home/webspam-2011-20datasets/uk-2011-web-spam-dataset

[72] H. A. Wahsheh, M. N. Al-Kabi, and I. M. Alsmadi, "A link and content hybrid approach for Arabic web spam detection," *Int. J. Intell. Syst. Appl.*, vol. 5, no. 1, pp. 30–43, Dec. 2012.

[73] M. Alsaleh and A. Alarifi, "Analysis of web spam for non-English content: Toward more effective language-based classifiers," *PLoS ONE*, vol. 11, no. 11, pp. 1–25, 2016.

[74] A. Alarifi and M. Alsaleh, "Web spam: A study of the page language effect on the spam detection features," in *Proc. 11th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2012, pp. 216–221.

[75] Y. Kaya and Ö. F. Ertuğrul, "A novel approach for spam email detection based on shifted binary patterns," *Secur. Commun. Netw.*, vol. 9, no. 10, pp. 1216–1225, Jul. 2016.

[76] O. Oludare, O. Stephen, O. Ayodele, and F. Temitayo, "An optimized feature selection technique for email classification," *Int. J. Sci. Technol. Res.*, vol. 3, no. 10, pp. 286–293, 2014.

[77] Z. Razi and S. A. Asghari, "Providing an improved feature extraction method for spam detection based on genetic algorithm in an immune system," *J. Knowl.-Based Eng. Innov.*, vol. 3, no. 8, pp. 569–605, 2017.

[78] H. Faris, A. M. Al-Zoubi, A. A. Heidari, I. M. M. Aljarah, M. A. Hassonah, and H. Fujita, "An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks," *Inf. Fusion*, vol. 48, pp. 67–83, Aug. 2019.

[79] H. Mohmmadzadeh and F. S. Gharehchopogh, "A novel hybrid whale optimization algorithm with flower pollination algorithm for feature selection: Case study email spam detection," *Comput. Intell.*, vol. 37, pp. 1–28, Jan. 2020.

[80] K. Neighbors, "A novel hybrid approach for email spam detection based on scatter search algorithm and K-nearest neighbors," *J. Adv. Comput. Eng. Technol.*, vol. 5, no. 3, pp. 169–178, 2019.

[81] M. Shuaib, O. Osho, I. Ismaila, and J. K. Alhassan, "Comparative analysis of classification algorithms for email spam detection," *Int. J. Comput. Netw. Inf. Secur.*, vol. 10, no. 1, pp. 60–67, Aug. 2018.

[82] O. E. Taylor, P. Harcourt, and P. Harcourt, "A model to detect spam email using support vector classifier and random forest classifier," *Int. J. Comput. Sci. Math. Theory*, vol. 6, no. 1, pp. 1–11, 2020.

[83] R. M. A. Mohammad, "An improved multi-class classification algorithm based on association classification approach and its application to spam emails," *IAENG Int. J. Comput. Sci.*, vol. 47, no. 2, pp. 187–198, 2020.

SANAA A. A. GHALEB received the bachelor's degree from the University of Aden, Yemen, in 2011, and the master's degree from Universiti Sains Malaysia, Malaysia, in 2017. She is currently pursuing the Ph.D. degree with the Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin. Her general research interests include technology-enhanced learning and instructional design and technology. Her research interests include computer networks and information security, cybersecurity, machine learning, artificial intelligence, swarm intelligence, and metaheuristic.

MUMTAZIMAH MOHAMAD received the bachelor's degree in information technology from Universiti Kebangsaan Malaysia, in 2000, the master's degree in computer science from Universiti Putra Malaysia, in 2005, and the Ph.D. degree from Universiti Malaysia Terengganu. She is currently an Associate Professor and the Deputy Dean of Academic and Postgraduate with the Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Malaysia. She has published more than 50 research articles, including book chapters and proceedings. She has worked as a researcher in several national funded research and development projects. Her research interests include machine learning, pattern recognition, artificial intelligence, and parallel processing.

SYED ABDULLAH FADZLI received the BMIS degree in management information system from Universiti Islam Malaysia, in 2002, the M.Sc. degree in software engineering from University Technologi Malaysia, in 2005, and the Ph.D. degree in computational engineering from Cardiff University, in 2013. His research interests include information, computer, and communication technology, software engineering, knowledge management, knowledge representation and machine learning, and embedded systems.

WAHEED ALI H. M. GHANEM received the B.Sc. degree in computer sciences and engineering from the University of Aden, Yemen, in 2003, and the M.Sc. degree in computer science and the Ph.D. degree in network and communication protocols from Universiti Sains Malaysia, in 2013 and 2019, respectively. His research interests include computer networks and information security, cybersecurity, machine learning, artificial intelligence, swarm intelligence, metaheuristic, and information technology.